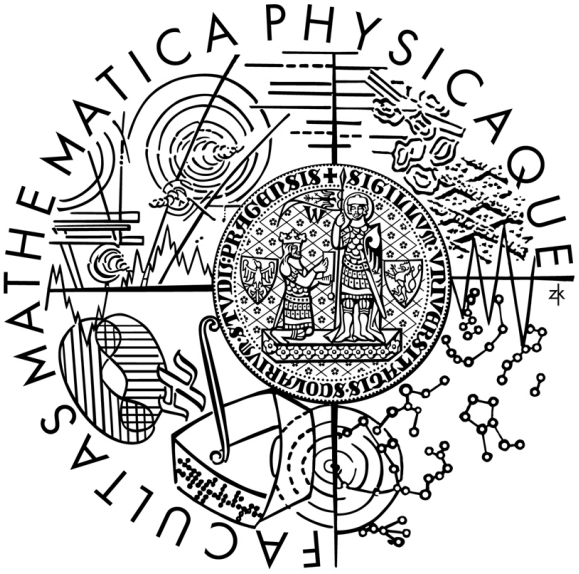
Univerzita Karlova v Praze

Matematicko-fyzikální fakulta

BAKALÁRSKA PRÁCA



Tomáš Smotrila

Simulace chování řidičů ve městě

Katedra aplikovanej matematiky

Vedúci bakalárskej práce: Mgr. Robert Babilon

Katedra aplikovanej matematiky

Študijný program: Informatika, Programovanie

2008

Týmto by som sa rád poďakoval Mgr. Robertovi Babilonovi za odborné vedenie mojej práce, venovaný čas a taktiež za jeho cenné rady a pripomienky. Ďalej by som sa chcel poďakovať rodine a všetkým priateľom, ktorí mi nezištne pomáhali otestovať softvér vyvinutý v tejto práci.

Prehlasujem, že som svoju bakalársku prácu napísal samostatne a výhradne s použitím citovaných prameňov. Súhlasím so zapožičiavaním práce.

V Prahe dňa 27.5.2008 Tomáš Smotrila

OBSAH

Název práce: Simulácia správania sa vodičov v meste

Autor: Tomáš Smotrila

Katedra (ústav): Katedra aplikovanej matematiky

Vedúci bakalárskej práce: Mgr. Robert Babilon, katedra aplikovanej matematiky

e-mail vedúceho: babilon@kam.ms.mff.cuni.cz

Abstrakt: Cieľom projektu je program simulujúci mestskú premávku a správanie sa vodičov v nej. Užívateľ má možnosť stavať cesty a zástavbu (domy, továrne, obchody, apod.). Výstupom je pohyb vodičov v meste postavenom užívateľom. Križovatky sú dvoch druhov: s hlavnou cestou a svetelná. Svetelné križovatky sú programovateľné užívateľom tak, aby mohol napr. vytvárať zelené vlny. Každý vodič sa pohybuje po meste na základe dopredu zvolenej stratégie, tie sú niekoľkých typov (napr. vodič jazdiaci vždy najkratšou trasou, vodič, ktorý sa rozhoduje podľa momentálnej situácie apod.) Jedným zo zmyslov projektu je, aby užívateľ bol schopný porovnať výhodnosť jednotlivých stratégií v cestnej sieti, ktorú postaví.

Kľúčové slová: simulácia, vodič, mestská premávka

Title: Simulation of Driver’s Behaviour

Author: Tomáš Smotrila

Department: Department of Applied Mathematics

Supervisor: Mgr. Robert Babilon, Department of Applied Mathematics

Supervisor's e-mail address: babilon@kam.ms.mff.cuni.cz

Abstract: The aim of the project is application simulating road traffic in town and driver’s behaviour in it. User is able to build roads and buildings (houses, factories, shops, etc). Application output is driver’s movement in town built by user. There are two types of crossroads: with main road and with traffic lights. Traffic lights are programmable by user in way to enable creation of green waves. Every driver performs his movement through the city based on pre-set strategy; these are several types (i.e. driver using the shortest path at every circumstances, driver, who bases his decisions on momentary situation etc.) One of project’s ideas is enabling user to compare effectiveness of particular strategies in road network he will build.

Keywords: simulation, driver, town traffic

1. Úvod

Hlavný podnet pre túto prácu pochádza z prostredia pražských ciest. Je ním cestná premávka v Prahe počas dopravnej špičky. Počas pracovných dní vo väčšine hlavných miest dosahuje premávka na cestách veľkú intenzitu v predpovedateľnej dobe kvôli veľkému počtu vozidiel používajúcich cesty v rovnaký čas. Tento fenomén sa nazýva dopravná špička (anglicky rush hour) [1].Obyvatelia mesta cestujúci ráno z domu do práce alebo poobede z práce domov tak ostávajú niekedy aj dlhé hodiny v dopravnej zápche.

Tento projekt má za cieľ skúmať vlastnosti cestnej premávky počas dopravnej špičky. Hlavným cieľom je vyvinutie programu, ktorý (zjednodušene) simuluje cestnú premávku a správanie sa vodičov v nej. Program umožní skúmať, ktoré správanie sa vodiča (voľba cesty mestom) je počas špičky najúspešnejšie.

Ďalším zo zmyslov práce je snaha nájsť vhodnú cestnú sieť, ktorá by výskyt dopravnej špičky minimalizovala. Užívateľ programu bude môcť nastavovaním smeru križovatiek s hlavnou cestou a programovaním logiky svetelných križovatiek skúmať, aký majú jeho zmeny vplyv na obmedzenie dopravnej špičky. V ideálnom prípade sa mu môže podariť vytvorenie takzvanej „zelenej vlny“, kedy sa pri správnom nastavení sústava nadväzujúcich križovatiek správa ako hlavná cesta.

Grafická podoba aplikácie je inšpirovaná hrou SimCity 3000, ktorá sa okrem iného tiež venovala cestovaniu obyvateľov v meste.

2. Analýza

2.1. Simluačný model

Pri analýze zadanie bola voľba simulačného modelu prvým krokom. Tá sa v prípade tohto programu odvíjala od viacerých faktorov. Išlo hlavne o to, aby sa automobily zobrazovali a pohybovali po cestách tak ako v skutočnosti, ale aby simulačný model bol, pokiaľ možno, čo najjednoduchší. Na výber sa núkali dva modely. Prvý, kde objekt má vlastnosti ako poloha či rýchlosť, ale chápeme ako hmotný bod, ktorý nemá svoje rozmery. Druhý, kde objekt má navyše aj svoje vonkajšie rozmery a tvar.
Variant s jednorozmernými objektmi je síce z pohľadu modelovania jednoduchší, avšak jeho grafické zobrazenie tak, aby odpovedalo dvojrozmernej skutočnosti, by bolo náročné. Preto je zvolená druhá možnosť s tým, že pohyb automobilov je vymedzený len vo vodorovnom alebo zvislom smere, čím sa značne zjednoduší simulačný model.
Pre reprezentáciu mesta je preto vhodné zvoliť štvorcovú sieť, v ktorej sa vodorovný alebo zvislý smer ľahko definuje. Jednotlivé objekty (budovy a cesty) navyše môžu byť uložené do dvojrozmerného poľa a manipulácia s nimi bude jednoduchšia. Táto reprezentácia je tiež výhodou pri výstavbe ciest a budov, kde bude užívateľovi ponúknutý celkom jednoduchý a pohodlný spôsob tvorenia zástavby kliknutím do políčka na mape.
Vzhľadom k tomu, že simulácia cestnej dopravy si vyžaduje interakciu objektov, ktorých pohyb nemožno predvídať dopredu, sa nedá zvoliť diskrétna simulácia a musí byť zvolená simulácia spojitá. Každý objekt teda simuluje svoju činnosť po elementárnych krokoch.

2.2. Definícia základných pojmov

Ďalším krokom v analýze je zadefinovanie pojmov, s ktorými sa bude narábať v programe, jeho implementácii a dokumentácii.

1. **Mesto** je základná entita, ktorá umožňuje riadiť simuláciu. Obsahuje zoznam **automobilov**, ktoré sa v ňom nachádzajú a **mapu mesta**.
2. **Mapa mesta** je štvorcová sieť pozostávajúca z **políčok**.
3. **Políčko** je základná bunka mapy, pokrývajúca jeden jej štvorček. Políčko môže byť typu
    a. **zem**
    b. **voda**
4. **Zóna** je políčko typu zem, na ktorom sa nachádza budova. Môže v nej bývať alebo pracovať obmedzené množstvo ľudí. Zóna môže byť typu
    a. **obytná**
    b. **obchodná**
    c. **industriálna**
5. **Cesta** je políčko typu zem alebo voda, na ktorom sa nachádza cesta a po ktorej budú jazdiť automobily. Cesta môže byť
    a. **slepá ulička**
    b. **rovná**
    c. **zákruta**
    d. **križovatka**
        i. **s dopravným značením,** s prednosťou na hlavnej ceste
        ii. **svetelná,** ktorej algoritmus bude programovateľný užívateľom
6. **Automobil**, označovaný aj ako **auto, človek, vodič** alebo **obyvateľ**, je objekt s vlastnou

logikou, ktorý sa pohybuje po meste.

2.3. Cestná sieť

Cestnú sieť v modeli môžeme definovať ako množinu políčok mapy kde je sa nachádza cesta. Za suseda políčka považujeme políčko ležiace buď nad, napravo, pod, alebo naľavo vedľa neho. Cestnú sieť podľa počtu a umiestnenia susedov typu cesta môžeme rozdeliť do štyroch kategórií:

1. **slepá ulička** - 1 susedná cesta
2. **rovná cesta** - 2 susedné cesty, obe ležiace v rovnakom stĺpci alebo riadku mapy
3. **zákruta** - 2 susedné cesty, obe ležiace na rôznych stĺpcoch a riadkoch mapy
4. **križovatka**
    a. tvaru písmena „T“ - 3 susedné cesty
    b. tvaru písmena „X“ – 4 susedné cesty

Poznámka : cesta s 0 susedmi typu cesta nie je definovaná, pretože sa predpokladá súvislá cestná sieť (viď ďalej) s aspoň dvoma políčkami.

Uzol cestnej siete chápeme ako cestu, ktorá je buď slepá ulička, zákruta alebo križovatka.
Z hľadiska grafových algoritmov je **cestná sieť** v tomto programe definovaná ako graf (neorientovaný), kde množina vrcholov pozostáva z uzlov cestnej siete a množina hrán pozostáva z (množín) rovných ciest. Základným invariantom v tomto programe je, že cestná sieť je vždy súvislý graf. Ďalším dôležitým invariantom je, že nikdy nesmú byt umiestnené dve križovatky vedľa seba, čo by skomplikovalo pohyb automobilu po meste.
Jednoduchá cesta je spoločné označenie pre slepú uličku a rovnú cestu. Len na jednoduchú cestu môžu automobily vstúpiť pri vychádzaní z budovy a môžu z nej zísť pri schádzaní z cesty do budovy.
Trasa automobilu mestom je chápaná ako usporiadaný zoznam (začiatok, cesta v grafe cestnej siete, koniec), kde začiatok je jednoduchá cesta, na ktorú automobil vychádza zo svojho domu a koniec je jednoduchá cesta, kde automobil schádza z cesty do domu.
Smer z nejakého uzlu cestnej siete je definovaný ako uzol cestnej siete, ktorý je s ním spojený hranou. Automobily sa tak v simulácii pohybujú po hranách cestnej siete vždy od jedného uzlu smerom k druhému.
Práve v uzloch cestnej siete sa uchovávajú informácie o polohe automobilov voči sebe navzájom. Každý uzol cestnej siete má zoznam smerov, z ktorých k nemu prichádzajú automobily. Pre každý smer má frontu automobilov, ktoré sa k nemu práve pohybujú.

2.4. Vodič

Človek žijúci v meste musí mať domov a prácu. Inak by simulovanie jeho správania sa stratilo zmysel, keďže by nemusel cestovať do práce a naspäť domov. Práca začína priemerne v tretine simulačného dňa a trvá presne tretinu simulačného dňa. Pre väčšiu rovnomernosť objemu cestnej premávky popoludní bolo človeku pridané chodenie do obchodu po práci. Nakupovanie trvá jednu hodinu.
Človek vykonáva svoj denný cyklus podľa nasledujúceho vzorca:

1. Prvý deň života v meste vyrazí do práce veľmi skoro, pretože ešte nepozná situáciu na ceste, a tak si dá väčšiu časovú rezervu, aby prišiel do práce načas.
2. Po práci vždy chodí nakupovať do rovnakého obchodu, ak nejaký má.
3. Po nákupe sa vracia naspäť domov.
4. Ak nemá obchod vracia sa po práci priamo domov.

5. Ak už človek absolvoval prvý deň v práci, ďalšie dni bude chodiť do práce podľa zmeraného času z prvého dňa s pridaním malej časovej rezervy.
6. Ak zameška do práce, ďalší deň sa do nej vydá skôr.

2.4.1. Správanie sa vodiča

Ako správanie sa (logiku) vodiča definujeme spôsob, akým sa snaží dostať do svojho cieľa načas, teda akým spôsobom volí svoju trasu po meste. Definujeme tieto 3 typy správania sa:

Konzervatívne správanie sa
Vodič v prvý deň, keď ešte nemá nastavený plán trasy, ho vytvorí počas prechádzania mestom tak, že na každej križovatke zvolí smer, ktorým vedie do cieľa najkratšia cesta. Po prvom dni už tento plán trasy nikdy nemení, okrem prípadov zmeny cestnej siete užívateľom.

Progresívne správanie sa
Definujme **hladinu tolerancie objemu** ako percentuálnu naplnenosť fronty automobilov, ktorá je nemenným atribútom progresívnej logiky vodiča. Ďalším nemenným atribútom tohto správania sa je **hladina tolerancie vzdialenosti**, ktorá predstavuje percentuálne predĺženie alternatívnej cesty oproti minimálnej. Vodič sa snaží pred križovatkou preplánovať svoj plán, ak vidí, že cesta ktorou je do cieľa najkratšie má vyššiu naplnenosť ako je jeho hladina tolerancie objemu. V tom prípade volí najkratšiu cestu z tých, ktoré majú menšiu naplnenosť a zároveň nie sú percentuálne dlhšie oproti minimálnej ako jeho hladina tolerancie vzdialenosti. Znamená to teda, že progresívny vodič je ochotný ísť inou cestou, len ak by si svoju cestu do cieľa výrazne nepredĺžil.

Chaotické správanie sa
Vodič na každej križovatke okrem cieľovej volí ako ďalší smer ten, ktorý má najmenší súčin obsadenosti cesty a vzdialenosti do cieľa.

2.4.2. Výpočet minimálnych ciest

Prvou otázkou pri výbere trasy automobilu mestom je informácia o dĺžke najkratšej cesty do cieľa použitím nejakého vrcholu. Odpoveď na túto otázku musí byť počas simulačného procesu dostupná v konštantnom čase, aby bola simulácia čo najplynulejšia. Keďže neexistuje žiaden algoritmus na výpočet minimálnej cesty s konštantnou časovou zložitosťou, musí byť výpočet minimálnych ciest vykonaný pred začatím simulácie.
Voľba algoritmu na výpočet ciest sa odvíja od správaní sa vodičov. Na výber sa núkajú Dijkstrov algoritmus s kvadratickou zložitosťou, ktorý nájde minimálne cesty z jedného vrcholu do všetkých ostatných vrcholov grafu a Bellman-Fordov algoritmus s kubickou zložitosťou, ktorý nájde najkratšie cesty medzi každými dvoma uzlami cestnej siete [2].
Pri konzervatívnom správaní sa, ktoré si vystačí s jednou cestou, by stačilo použiť Dijkstrov algoritmus, ale pri progresívnom a chaotickom už musíme vedieť vzdialenosti náhodných (a teda všetkých možných) dvojíc.
Minimálne cesty medzi každými dvoma vrcholmi vieme zistiť zavolaním Dijkstrovho algoritmu na každý vrchol, ale z implementačného hľadiska je Bellman-Fordov algoritmus z troma cyklami a jednoduchou podmienkou elegantnejšie riešenie.
Po každej zmene cestnej siete je treba znovu vyhľadať najkratšie cesty.

2.4.3. Pravidlá cestnej premávky

Všetky automobily v simulácii sa musia riadiť jednotnými pravidlami cestnej premávky.

1. Automobil vychádza z budovy alebo schádza do budovy len z jednoduchej cesty. Nikdy nemôže vstúpiť alebo opustiť cestnú sieť v križovatke alebo zákrute.
2. Automobil vyjde na jednoduchú cestu (a jazdný pruh pre svoj ďalší smer) len v prípade, že by nevošiel do brzdnej dráhy iného automobilu.
3. Na ceste bude dodržiavať maximálnu povolenú rýchlosť.
4. Spomalí pri prechode zákrutou.
5. Na svetelnej križovatke sa bude riadiť svetelnými signálmi.
6. Na križovatke s hlavnou cestou sa riadi dopravným značením.
7. Prispôsobí svoju rýchlosť automobilu pred ním tak, aby medzi nimi nedošlo ku kolízii.

2.4.4. Pohyb automobilu mestom

Automobil svoj pohyb simuluje po elementárnych krokoch. Základnými údajmi o pohybe automobilu sú:

- **Poloha** – vektor reálnych čísel, poloha horného ľavého rohu objektu
- **Rýchlosť –** vektor reálnych čísel

Parametre automobilu sú :

- **maximálna konštrukčná rýchlosť automobilu**
- **akceleračná sila –** udáva prírastok rýchlosti za jeden elementárny krok
- **brzdná sila -** udáva úbytok rýchlosti za jeden elementárny krok

V každom kroku sa automobil rozhodne o akú hodnotu zmení svoju rýchlosť. Môže buď pridávať, vtedy sa jeho k rýchlosti pripočíta konštanta akceleračná sila. Ak automobil brzdí od rýchlosti odčíta konštantu brzdná sila, alebo inú hodnotu ktorá je menšia.
Maximálna rýchlosť je minimum z maximálnej konštrukčnej rýchlosti automobilu a rýchlostného limitu na cestách.
Aktuálny cieľ automobilu je bod na mape určený uzlom cestnej siete, ku ktorému automobil práve smeruje. Pokiaľ sa automobil nachádza na ceste, stále smeruje k nejakému uzlu cestnej siete, a tak má aktuálny cieľ vždy.
Bod zastavenia je bod, na ktorom automobil zastaví, ak začne okamžite a naplno brzdiť.
Pri pohybe automobilu k aktuálnemu cieľu musí vždy platiť nasledovný invariant : bod zastavenia sa nachádza pred aktuálnym cieľom, alebo je s ním totožný.

Pravidlá pre pohyb automobilu sú nasledovné :

1. Ak zistí, že nebrzdením v aktuálnom kroku, by v ďalšom neplatil invariant začína brzdiť.
2. Ak je jeho rýchlosť takmer minimálna (o niečo málo vyššia ako brzdná sila alebo je nulová) a je v tesnej blízkosti aktuálneho cieľa presunie sa do neho.
3. Automobil sa zo státia rozbieha do maximálnej povolenej rýchlosti. Akceleruje pri tom naplno za dodržiavania spomínaného invariantu.
4. Pri dosiahnutí aktuálneho cieľa si nastaví nový podľa ďalšieho uzlu cestnej siete, ktorý nasleduje na trase.

Ďalšou skutočnosťou, ktorú treba zohľadniť je, aby automobil nenaraził do automobilu, ktorý sa nachádza pred ním. Automobil sa najprv rozhodne bez ohľadu na okolité automobily o tom, akú zmenu svojej rýchlosti chce vykonať na základe pravidiel 1.-4. a až tak sa presvedčí či mu v tomto rozhodnutí nebráni automobil nachádzajúci sa pred ním.
V aktuálnom elementárnom kroku nevie automobil, či vozidlo pred ním už vykonalo rozhodnutie alebo nie. Preto pri zrýchľovaní alebo udržiavaní rýchlosti sa presvedčí, že aj v najhoršom prípade, kedy automobil pred ním ešte nespravil svoje rozhodnutie, ale v aktuálnom kroku bude brzdiť, nebude jeho bod zastavenia pred bodom zastavenia automobilu idúceho pred ním. Ak by náhodou mohol nastať tento prípad začne automobil brzdiť, aby si udržal bezpečný odstup od vozidla idúceho pred ním.

2.5. Svetelná križovatka

Pri príchode ku svetelnej križovatke sa automobil riadi nasledujúcimi pravidlami:

1. Ak má červenú pre svoj smer, spomaľuje až dosiahne najbližšiu možnú vzdialenosť ku hranici križovatky (hranica križovatky alebo posledný automobil smerujúci k nej). Ďalej čaká na zelenú pre svoj smer a poprípade sa posunie bližšie dopredu, ak sa pred ním uvoľní priestor.
2. Ak má zelenú pre svoj smer, prejde križovatkou ako cez zákrutu alebo rovnú cestu. Urobiť tak môže, jedine ak cesta za križovatkou, kam smeruje, je voľná. Tak má zaručený bezpečný výjazd z križovatky.

Logika svetelnej križovatky pracuje s tabuľkou krokov. Tá obsahuje vždy 4 kroky. Každý **krok** je definovaný ako dvojica pozostávajúca z **dĺžky kroku** a **zoznamu smerov**, ktorými môžu automobily prechádzať križovatkou v danom kroku. **Smer prechodu križovatkou** je definovaný ako usporiadaná dvojica uzlov cestnej siete (**A,B**), kde **A** je uzol cestnej siete odkiaľ automobil vstupuje do križovatky a **B** je smer ktorým ju opúšťa. Svetelná križovatka má ešte ďalšie parametre:

- **clockwise** - údaj o tom, či sa kroky menia vzostupne alebo zostupne
- **first_state** – ktorým korkom má logika začínať svoj algoritmus.

2.5.1. Popis fungovania logiky križovatky

Na začiatku sa nastaví aktuálny krok na prvý s nenulovou dĺžkou. Ak taký neexistuje, aktuálny krok bude nastavený na prvý krok v tabuľke, a križovatka sa tak stáva nepriechodnou. Po uplynutí času trvania kroku a **bezpečnostnej časovej rezervy**, ktorá slúži na zabezpečenie vyprázdnenia križovatky od automobilov, sa podľa premennej clockwise nastaví aktuálny krok na ďalší s nenulovým časom, ak taký existuje. Inak sa aktuálny krok sa nemení. Kroky križovatky sa menia cyklicky, teda ak je aktuálny krok posledný, tak za ním nasleduje prvý (ak je parameter clockwise nastavený na true).
V programe sa prednastavene vytvára základná logika križovatky, ktorá je charakterizovaná rovnakým prístupom ku každému smeru. To znamená, že v každom kroku je umožnený prejazd automobilom z jedného smeru do všetkých ostatných. Tieto smery sú obmieňané v rovnakých časových intervaloch, a preto je vhodné použitie takejto križovatky na miestach, kde zo všetkých strán prichádzajú automobily v rovnakom pomere.

2.5.2. Synchronizácia svetelných križovatiek

Definujme konštantu, **časová dĺžka políčka** ako čas, za ktorý prejde automobil jedno políčko pri plnej rýchlosti. Synchronizácia svetelných križovatiek spočíva v nasledovnej úvahe: užívateľ bude nastavovať trvanie jedného kroku križovatky po násobkoch tejto konštanty. Pre dosiahnutie zelenej vlny je potrebné, aby automobily smerujúce z najbližšej križovatky dorazili k aktuálnej najskôr vo chvíli, keď sa začne krok, v ktorom majú zelenú. Nastavením dĺžky predošlého kroku tak, aby sa končil v čase, kedy dorazia automobily z vedľajšej križovatky (násobok časovej dĺžky políčka, ktorý odpovedá počtu políčok medzi synchronizovanými križovatkami) sa dosiahne efekt zelenej vlny.

2.6. Križovatka s hlavnou cestou

Pre naimplementovanie hlavnej cesty, tak aby presne zodpovedala skutočným pravidlám cestnej premávky by sa mohlo stať, že simulačný proces by sa veľmi spomalil kvôli nadmernému množstvu výpočtov.
Príčinou je fakt, že v tejto simulácii sa vyskytuje až 24 križovatiek s hlavnou cestou a počet možností ako prejsť hlavnou cestou je tiež vysoký (12). Z toho vyplýva, že by muselo byť ošetrené veľké množstvo (288) prípadov, konkrétneho prejazdu konkrétnou križovatkou. Preto bolo použité nasledujúce zjednodušenie:

Pri príchode automobilov ku križovatke s hlavnou cestou je im umožnený prechod križovatkou v tomto poradí:

1. Ak ide automobil po hlavnej ceste, je mu vždy umožnený prejazd
2. Ak schádza z hlavnej cesty tak, že križuje protiidúci hlavný smer, musí byť tento protismer voľný.
3. Ak prichádzajú naraz automobily po rovnej hlavnej ceste tak, že z nej oba chcú schádzať, je im obom umožnený prejazd naraz.
4. Ak prichádza z vedľajšej cesty s **vyššou prioritou**, musia byť voľné oba hlavné smery.
5. Ak prichádza z vedľajšej cesty s **nižšou prioritou**, musia byť všetky ostatné smery voľné.

Prioritu vedľajšej cesty má zmysel rozlišovať len v prípade križovatky tvaru písmena „X“, ktorá má dve vedľajšie cesty. Vyššia priorita vedľajšej cesty je určená podľa jej polohy vzhľadom ku križovatke v nasledujúcom poradí:

1. Vedľajšia cesta, nachádzajúca sa nad križovatkou
2. Vedľajšia cesta, nachádzajúca sa vpravo križovatkou
3. Vedľajšia cesta, nachádzajúca sa pod križovatkou
4. Vedľajšia cesta, nachádzajúca sa vľavo križovatkou

Pravidlá č.4. a č.5. spôsobia, že automobily prichádzajúce z vedľajšej cesty neprejdú križovatkou ak po nej už idú automobily, ktoré im vôbec nekrižujú smer. Podľa skutočných pravidiel cestnej premávky (a aj v simulácii) by tak reálne mohli učiniť bez toho, aby došlo ku kolízii, ale tieto pravidlá im to zakazujú.
Toto zjednodušenie ide však na úkor jednoduchosti implementácie hlavnej cesty a zníženia množstva výpočtov pri testovaní možnosti prejazdu križovatkou. Pôvodný účel križovatky s hlavnou cestou, ktorým je pravidlo č.1, je tak zachovaný. Nadefinovaním hlavných ciest sa urýchli tok cestnou sieťou v danom smere, čo prospieva jej priepustnosti.

3. Programová dokumentácia

3.1. Technológie

Ako cieľová platforma projektu bol zvolený operačný systém Windows. Na realizáciu projektu je zvolený objektový jazyk Managed Extensions pre C++, ktorý umožní v projekte použiť moderné objekty .NET frameworku 2.0 ako napríklad dvojito buffrovaná grafika, za použitia známej syntaxe jazyka C++. Keďže program využíva .NET framework bude spustiteľný na Windows Vista bez akejkoľvek nutnosti ďalšej inštalácie, a spustiť sa bude dať aj na operačných systémoch počínajúc Windows 98 a vyššie s nainštalovaným .NET framework 2.0.
Managed Extensions pre C++, označované ako aj Managed C++ je sada derivácií z jazyka C++ pochádzajúca od Microsoftu, zahŕňajúca gramatické a syntaktické rozšírenia, kľúčové slová a atribúty, ktorá má priniesť C++ syntax a jazyk do .NET Framework. Tieto rozšírenia umožňujú použiť C++ kód v Common Language Runtime (CLR) vo forme manažovaného kódu a tiež pokračovať v súčinnosti s pôvodným kódom. Managed C++ nie je samostatným jazykom ani plne kvalifikovaným jazykom [3].
V projekte sú použité dva priestory mien a to sim a RushHour. Sim obsahuje kód simulácie a je napísaný v jazyku C++, čo znamená, že je prenositeľný aj na iné platformy ako napríklad Unix. Priestor RushHour definuje grafické užívateľské rozhranie je napísaný v Managed C++.

3.2. Dátové štruktúry

3.2.1. Zoznam tried použitých v projekte

Triedy zúčastňujúce sa simulácie:

`sim::Town`	Mesto, v ktorom sa odohráva simulácia
`sim::Object`	Abstraktný predok všetkých ostatných tried v objektovom modeli
`sim::Automobile`	Automobil/človek, ktorý ho riadi
`sim::Field`	Štvorček na mape mesta
`sim::Zone`	Zóna (rezidentálna, komerciálna, industriálna)
`sim::Road`	Cesta ľubovoľného typu
`sim::Crossroad`	Križovatka
`sim::SimpleRoad`	Cesta bez zákruty
`sim::StraightRoad`	Rovná cesta
`sim::CurvedRoad`	Zákruta/ohyb
`sim::DeadEnd`	Slepá ulička

Pomocné triedy:

`sim::CarQueue`	Špeciálna fronta áut, s vkladaním a vyberaním z prostriedku
`sim::CrossroadLogic`	Logika semaforu
`sim::Director`	Trieda, ktorá v sebe uchováva grafové informácie cestnej siete
`sim::RoutePlan`	Plán cesty automobilu mestom
`sim::tester`	Štruktúra, ktorá sa používa na testovanie

Triedy definujúce okná programu:

`RushHour::Form1`	Hlavné okno programu
`RushHour::FormBehaviourComparison`	Okno, s porovnaním správaní sa vodičov

3.2.2. Town

Trieda **Town** má v prvom rade za úlohu poskytnúť, riadenie simulácie, ďalej načítanie, menenie a ukladanie mapy a v neposlednom rade sprostredkovanie štatistických dát zozbieraných počas simulácie. Uchováva v sebe :

- mapu mesta
- vektor automobilov
- grafové dáta cestnej siete
- sadu testovacích máp
- simulačný čas
- ďalšie pomocné premenné

Načítanie a ukladanie mapy

Mapa mesta je uložená v dvojrozmernom poli, kde na každej pozícii sa nachádza objekt typu **`field`**. Mapu môže byť načítaná z troch rôznych formátov :

- **`internal_map_input`** metódou **`LoadMapFromInternal`**
- textový súbor metódou **`LoadMapFromTextFile`**
- binárny súbor metódou **`LoadMapFromBinFile`**

Ak je aplikácia v testovacom režime, načíta sa mapa poskytnutá privátnym členom **`tester_`** vo formáte **`internal_map_input`**. Tieto formáty v sebe neuchovávajú informácie o automobiloch, ktoré sa vyskytujú v meste. Po načítavaní mapy musí mapa prejsť procesom zisťovania dát o cestnej sieti:

1. Preznačenie ID všetkých križovatiek tak, aby boli číslované od 0 zaradom pomocou metódy **`RemarkCrossroadsID`** Táto podmienka je dôležitá kvôli vypočítavaniu minimálnych ciest.
2. Načítanie informácií o cestnej sieti metódou **`DetermineCrossroadsData`** , kde sa zistí nielen graf cestnej siete, ale aj jej jednotlivé políčka obdržia potrebné informácie o svojich cestných susedoch.
3. Pre každé políčko zóny sa zistí najbližšia cesta metódou **`DetermineClosestRoads`** Po zistení potrebných údajov o cestnej sieti prichádza na rad počítanie grafových dát cestnej siete, ktoré neskôr pomôžu automobilom pri ich voľbe trasy mestom. Za týmto účelom bola vytvorená trieda **`Director`**, ktorá uchováva grafové dáta. V predošlom bode č.2 dostane táto trieda informácie o grafe cestnej siete a následne použije Bellman-Fordov algoritmus na vypočítanie minimálnych ciest pre každé dva uzly cestnej siete.

Generovanie automobilov

Na začiatku každého simulačného dňa sa generujú nové automobily v metóde **`Generator`**. Ak je aplikácia v testovacom režime načítajú sa automobily z testovacieho scenára, poskytnutého privátnym členom **`tester_`**. Počet novo vygenerovaných automobilov je nastaviteľný užívateľom alebo je možné ho ponechať na automatické generovanie. Na tento účel slúži metóda **`SetAutomaticGeneration`**. Automatické generovanie znamená, že sa vygeneruje polovica z

maximálnej možnej kapacity automobilov (každý automobil si nájde prácu aj domov). Pri ručnom zadávaní sa dá nastaviť, aby sa celé mesto vyprázdnilo alebo sa dá zaplniť do maximálnej možnej kapacity, pomocou metódy **`SetNewCars.`**

Simulácia

Po úspešnom načítaní mapy a vygenerovaní automobilov je všetko pripravené na začatie simulácie. Simulácia je vykonávaná za pomoci časovača, ktorý je umiestnený v hlavnom okne programu. Ten v pravidelných intervaloch volá metódu **`Step`**, ktorá vykoná jeden elementárny krok simulácie. Krok simulácie znamená vykonanie elementárnych krokov križovatiek a automobilov. Na ukončenie simulácie slúži metóda **`Sleep`**, ktorá zariadi, aby sa všetky automobily premiestnili do svojej východiskovej polohy pre ďalší deň, teda do svojho domu.

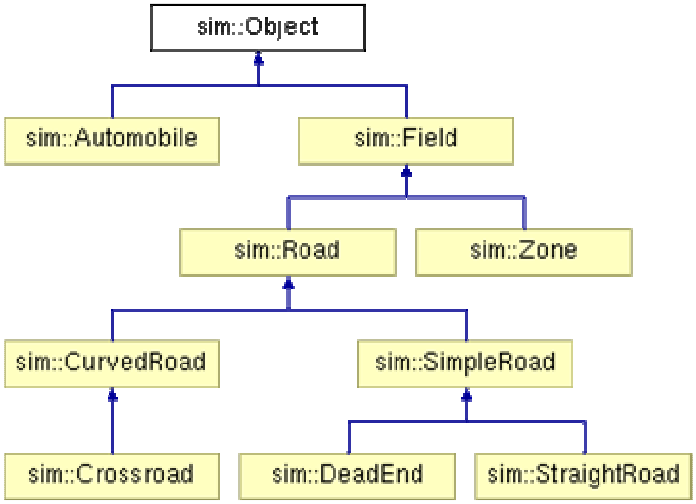
Úpravy mapy medzi simulačnými dňami

Pri pridávaní políčok typu **`Zone`** sa vykonajú úkony tak ako pri načítavaní mesta. Pri vymazaní tohto políčka je to však zložitejšie. Ak náhodou vymažeme políčko, kvôli ktorému sa stane nejaký automobil nezamestnaným treba ho prezamestnať. Ak ostane bez domova, treba automobil zmazať. Uvedené úkony zabezpečuje metóda **`ReSocializePopulation`**.

Pri zmene cestnej siete celá situácia ešte komplikovanejšia. Pridanie novej cesty môže ovplyvniť políčka zóny (zmena najbližšej cesty). Vymazanie cesty môže spôsobiť, že nejaká budova sa ocitne ďaleko od cestnej siete a tým pádom sa stane neobývateľná. Nastáva rovnaká situácia ako pri mazaní políčka typu **`Zone`**. Test, či môžeme odstrániť políčko cesty bez toho, aby sa cesta rozpadla na 2 nesúvislé komponenty testuje metóda **`CanRemoveRoadAt`** pomocou algoritmu vlny. Naviac po každej zmene cestnej siete treba znovu opraviť dáta o cestnej sieti a vypočítať minimálne cesty metódami **`PrepareRoadData`** a **`PrepareGraphData`**.

3.2.3. Object

Trieda **`Object`** je abstraktný predok všetkých tried, ktoré sa zúčastňujú simulačného procesu. Jej potomkovia sú automobil a políčka na mape. Kompletnú dedičnosť znázorňuje UML diagram na obrázku č.1.



Obrázok č.1 – UML diagram objektového modelu

Každý objekt v sebe nesie:

- typ inštancie objektu
- identifikátor objektu
- pozícia horného ľavého rohu objektu vo svete (dvojica reálnych čísel)
- či objekt hlási udalosti mestu

Základné metódy typu get, ktoré pracujú s uvedenými atribútmi objektu sú **`GetId`**, **`GetPosition`**, **`ObjectType`** a **`GetCoordinates`**, ktorá vracia súradnice mapy, na ktorých sa objekt nachádza

Jedinou metódou typu set je **`SetId`**. Z hľadiska nemennosti ID objektu by nemala byť definovaná a ID objektu by sa malo nastavovať iba pri vytváraní objektu. Avšak pri zmene cestnej siete a vypočítavaní trás je potrebné si prečíslovať uzly cestnej siete pomocou tejto metódy.

Polymorfizmus objektov umožňujú využívať metódy pretypovania na potomkov napr. **`access_automobile`**.

Najdôležitejšou metódou všetkých objektov ktoré dedia od triedy **`Object`** je metóda **`Step(const time_unit time)`**, ktorá definuje správanie sa objektu. Pomocou parametra time sa objekt dozvedá čas simulácie, na základe ktorého koná svoje rozhodnutia.

Na účely testovania a ladenia je potrebné priebežne sledovať stav niektorých vybraných objektov. Pomocou metódy **`SetReportMode`** sa dá nastaviť, aby objekt v každom kroku hlásil akcie, ktoré vykonáva. Preto každý objekt obsahovať sadu metód slúžiacich na výpis stavu objektu a jeho premenných na štandardný výstup. Metóda flush vypisuje stav objektu a je, dá sa povedať, odľahčeným ekvivalentom locals vo vývojom prostredí. a používa preťaženú metódu report na vypisovanie jednotlivých premenných. Vhodným umiestnením metódy report do zdrojového kódu sa dá napodobniť krokovanie.

3.2.4. Automobile

Základnou úlohou tejto triedy je simulovať cestnú premávku a zbierať štatistické údaje o svojej ceste, ktoré neskôr poslúžia na účely vyhodnotenia efektivity cestnej siete a porovnanie jednotlivých typov správaní sa vodičov. V dokumentácii je označovaná aj ako **človek**, **obyvateľ**, **vodič**, **automobil**, **autíčko**, **auto**.

Vytvorenie automobilu

Automobil v meste nemôže existovať samostatne, ale je nutná jeho spolupráca s ostatnými triedami vyskytujúcimi sa v simulácii. Preto sa jeho vytvorenie vykonáva v metóde **`Town::CreateCar`**, kde je jeho vytvorenie spojené s ďalšími úkonmi, ktoré treba pri jeho vytvorení vykonať. Pri vytváraní pomocou konštruktora **`Automobile(Town*,Zone*,DriversBehaviour)`** automobil v nej dostane pointre na mesto **`my_town_`**, svoj domov **`home_`** a svoju logiku **`behaviour_`**. Jeho ostatné parametre sú vzápätí nastavené metódami:

- **`SetReportMode`** - podľa testovacieho režimu zapne/vypne vypisovanie hlásení automobilu na obrazovku
- **`SetParameters`** - nastaví automobilu fyzikálne parametre
- **`SetWork`** - priradí automobilu prácu, kde bude zamestnaný
- **`SetShop`** - priradí automobilu obchod, kam bude po práci chodiť nakupovať

Simulovanie denného cyklu automobilu

Po vytvorení sa automobil nachádza doma a je pripravený na začatie simulácie. V každom kroku simulácie je zavolaná metóda **`Step`**, ktorá vykoná jeden elementárny krok automobilu. Ak je správny čas, snaží sa automobil vydať sa na cestu do práce, obchodu alebo naspäť domov pomocou metódy **`TryToStrikeOutOnRoute`**. Ak sa nachádza automobil na ceste tak sa snaží dostať sa do cieľa svojej cesty pomocou metódy **`FollowRouteTarget`**.

Pohyb automobilu mestom

Automobil sa po cestnej sieti môže pohybovať po rovnej ceste, zákrute, križovatke a slepej uličke. Trasa automobilu sa začína resp. končí na najbližšej ceste, ležiacej pri budove, z ktorej vychádza resp. do ktorej prichádza. Najčastejšie je to rovná cesta, menej často už slepá ulička. Typicky ďalej pokračuje po rovných cestách ku najbližšiemu uzlu cestnej siete, ktorými sú zákruta alebo križovatka pomocou metódy **`GoStraight`**. Ak sa už dostane blízko križovatky, použije metódu **`GoThroughCrossroad`** na prechod ňou. Ak je už v blízkosti zákruty použije metódu **`GoThroughCurve`**. Je treba ošetriť aj špeciálny prípad, kedy automobil nemusí cestovať žiadnym uzlom cestnej siete. Ďalší uzol cestnej siete ku ktorému sa má vydať získava automobil zavolaním metódy **`GetNextTarget,`** ktorá podľa typu správania sa automobilu vráti ďalší uzol cestnej siete, ktorým sa treba vydať. Tieto uzly si automobil v prvý deň ukladá do plánu cesty **`route_plan_`**. Nasledujúce simulačné dni vracia metóda **`GetNextTarget`** uzly, na základe správania sa a plánu cesty. Buď preplánuje aktuálny plán alebo sa ho bude držať.

Koniec života automobilu

Automobil je pred zničením vyradený z pomocných štruktúr mesta v metóde **`Town::DeleteCar.`** Nutnosť zničenia automobilu môže nastať :

- na žiadosť užívateľa (klikol na tlačidlo **Delete all cars**)
- pri vymazaní rezidentálnej zóny v ktorej automobil býva
- pri vymazaní cestnej siete tak, že sa zóna, kde žije ocitne ďaleko od cestnej siete a stane sa neobývateľnou

3.2.5. Field

Abstraktná trieda, ktorá nemá inštanciu. Je použitá ako základná stavebná jednotka mapy mesta. Rozširuje triedu Objekt o údaji typu políčka a svoje koordináty na mape.

3.2.6. Zone

Reprezentuje políčko, v ktorom sa môžu nachádzať obyvatelia mesta (automobily). Automobily putujú v meste z políčka zóny do iného. Zóna môže byť troch typov podľa účelu cesty k nej:

- rezidentálna, kde obyvatelia žijú, odkiaľ sa vydávajú do práce a kam sa vracajú večer z mesta
- industriálna, kde obyvatelia pracujú a odkiaľ po práci chodia na nákupy
- komerciálna, kam obyvatelia chodia nakupovať ale aj pracovať

Zo spomínaných účelov vyplýva, že trieda **`Zone`** musí rozširovať triedu **`Field`** o nasledujúce dátové položky:

- informácie o počte obyvateľov, ktorí v nej žijú
- celkovú kapacitu obyvateľov, ktorí sa do nej zmestia
- pointer na najbližšiu cestu, na ktorú budú z nej vychádzať resp. schádzať automobily
- čas, kedy sa začína v továrni/obchode pracovať

3.2.7. Road

Trieda **`Road`** reprezentuje políčko cesty ľubovoľného typu, po ktorom sa môže pohybovať automobil. Jej hlavnou úlohou je uchovávať informácie o polohe automobilov, ktoré sa po nej pohybujú a definovať rozhranie pre prácu s týmito informáciami. Implementácia rozhrania je však ponechaná na jej potomkov, pretože z ich rozdielnej povahy a účelu si každý vyžaduje inú implementáciu. Preto je táto trieda abstraktná. Triedu **`Field`** rozširuje o nasledujúce členy:

- zoznam priamych susedov typu cesta
- typ cesty, podľa potomka
    - rovná cesta
    - slepá ulička
    - zákruta
    - križovatka
- metóda **`AddDirection`**, ktorou ceste dávame informáciu o nejakom najbližšom uzle cestnej siete
- index obrázka konkrétnej cesty vo vektore obrázkov ciest

Uchovávavanie informácií o polohe automobilov

Pre rozumné uchovávanie informácií o polohe automobilov na cestách bol zvolený nasledovný model. Každý uzol cestnej siete má zoznam smerov (iný uzol cestnej siete), z ktorých k nemu môžu prichádzať automobily. Tieto automobily majú tú vlastnosť, že vždy idú za sebou v priamke a žiaden z nich sa neprebieha. Z toho vyplýva, že najlepšou štruktúrou na uchovanie tohto stavu bude fronta, ktorá ale musí zohľadniť nasledovné skutočnosti :

- Ak automobil vychádza na jednoduchú cestu z budovy, potrebuje sa do tejto fronty zaradiť na miesto rôzne od začiatku a konca. Za týmto účelom je dodefinovaná metóda Insert, ktorá vkladá automobil na určené miesto "uprostred" fronty. Volaniu tejto metódy však musí vždy predchádzať zistenie, či tak môžeme urobiť pomocou metódy **`QueueFreeAt.`**
- Niekedy je fronta automobilov plná, a preto sa do nej nesmie vkladať ďalší metódou **`Push.`** Na overenie, či môžeme automobil zaradiť do fronty slúži metóda **`CanPush`**.
- Na vyberanie automobilu z fronty metódou Pop nie sú kladené žiadne obmedzenia.

Pri vstupovaní automobilu do fronty smerujúcej k uzlu cestnej siete, si zistí automobil, aký je konkrétny bod kde sa do nej vstupuje zavolaním metódy **`GetEntryPoint`**. Metóda **`GetExitPoint`** má opačný význam.

3.2.8. Crossroad

Trieda **`Crossroad`** reprezentuje križovatku, ktorej hlavnou úlohou je riadiť premávku spôsobom

požadovaným užívateľom. Volaním metódy **GreenForDirectionFrom** automobil získava informáciu o tom, či ňou v danom časovom okamihu môže prejsť daným smerom, určeným dvojicou uzlov cestnej siete odkiaľ kam. Križovatka môže pracovať v dvoch režimoch:

- svetelná križovatka, ktorá je implementovaná pomocou triedy **CrossroadLogic**
- križovatka s hlavnou cestou, ktorej logika je definovaná priamo v metóde **GreenForDirectionFrom**

Za účelom nastavovania križovatky sú pridané metódy

- **SetMainStreet**, ktorá nastavuje smery hlavnej cesty pomocou zoznamu susedov
- **GetLogic**, ktorá poskytne pointer na logiku, ktorá je potom priamo nastavovaná v okne hlavnej aplikácie
- **SwitchCrossroadType**, ktorá prepína režim križovatky

3.2.9. SimpleRoad

Trieda **SimpleRoad** je abstraktná a rozširuje vlastnosti triedy **Road** o fakt, že len na tento typ cesty môžu automobily vyhádzať a schádzať z neho. Po jednoduchej ceste sa automobil môže pohybovať len rovno.

3.2.10. StraightRoad

Trieda **StraightRoad** reprezentuje rovnú cestu. Ako jediný potomok triedy **Road** neobsahuje fronty automobilov, ktoré k nej smerujú, pretože nie je uzlom cestnej siete.

3.2.11. CurvedRoad

Trieda **CurvedRoad** reprezentuje zákrutu a zaisťuje zmenu smeru pohybu automobilu z vodorovného do zvislého či naopak. Keďže táto trieda už nie je abstraktná definuje spoločnú štruktúru typu **Queues**, v ktorej sú uchovávané informácie o polohe automobilov vo frontách, aj pre svojho priameho potomka **Crossroad**.

3.2.12. DeadEnd

Trieda **DeadEnd** reprezentuje slepú uličku, teda políčko cesty, ktoré má jediného suseda typu cesta.

3.2.13. CarQueue

CarQueue reprezentuje špeciálnu frontu áut, s vkladaním a vyberaním z prostriedku. Je využívaná v uzloch cestnej siete teda **DeadEnd**, **CurvedRoad** a **Crossroad**. Nasledujúce metódy umožňujú :

- **`pushCar`** - vloženie automobilu na koniec fronty
- **`pushCarAt`** - vloženie automobilu "doprostriedka" na určené miesto fronty
- **`popCar`** - vyradenie automobilu zo začiatku fronty alebo z ktoréhokoľvek miesta fronty
- **`roadFreeAt`** - zistiť, či je fronta voľná v zadanom bode(najbližší automobil smerujúci k tomuto miestu je ďaleko) a môže sa tam teda vložiť automobil
- **`CanPush`** - zistiť, či sa môže automobil zaradiť na koniec fronty
- **`WillBeFreeSpace`** - zistiť, či vo fronte bude miesto pre zaradenie ďalšieho automobilu na koniec
- **`DirectionIsFree`** - zistiť križovatke, či sa nachádza prvý automobil vo fronte ďaleko od jej začiatku

3.2.14. CrossroadLogic

Hlavnou zodpovednosťou logiky semaforu je poskytnúť informáciu, či v stave, v ktorom sa nachádza, je možný prechod križovatkou v danom smere. Na to slúži metóda **`CanPassThrough`**. Logika svetelnej križovatky pracuje s tabuľkou krokov. Tá obsahuje vždy 4 kroky. Každý krok je definovaný ako dvojica pozostávajúca z dĺžky kroku a zoznamu smerov, ktorými môžu automobily prechádzať križovatkou v danom kroku. Smer je definovaný ako usporiadaná dvojica uzlov cestnej siete (A,B), kde A je uzol cestnej siete odkiaľ automobily môžu vstúpiť do križovatky a B je smer ktorým ju môžu opustiť. Svetelná križovatka má ešte ďalšie parametre :

- **`clockwise_`** - údaj o tom, či sa kroky menia vzostupne alebo zostupne
- **`first_state_`** – ktorým korkom má logika začínať svoj algoritmus.

Popis fungovania logiky križovatky

Na začiatku sa nastaví aktuálny krok na prvý s nenulovou dĺžkou. Ak taký neexistuje, aktuálny krok bude nastavený na prvý krok v tabuľke, a križovatka sa tak stáva nepriechodnou. Po uplynutí času trvania kroku a bezpečnostnej časovej rezervy, ktorá slúži na zabezpečenie vyprázdnenia križovatky od automobilov, sa podľa premennej clockwise nastaví aktuálny krok na ďalší s nenulovým časom, ak taký existuje. Inak sa aktuálny krok sa nemení. Kroky križovatky sa menia cyklicky, teda ak je aktuálny krok posledný, tak za ním nasleduje prvý (ak je parameter **`clockwise_`** nastavený na true). Prednastavene sa vytvára základná logika križovatky, ktorá je charakterizovaná rovnakým prístupom ku každému smeru. To znamená, že v každom kroku je umožnený prejazd automobilom z jedného smeru do všetkých ostatných. Tieto smery sú obmieňané v rovnakých časových intervaloch, a preto je vhodné použitie takejto križovatky na miestach, kde zo všetkých strán prichádzajú automobily v rovnakom pomere.

3.2.15. Director

Trieda, ktorá v sebe uchováva informácie o grafe cestnej siete a jej minimálnych cestách. Ako vrcholy v grafe sa chápu uzly cestnej siete, teda slepé uličky, zákruty a križovatky. Vrcholy však nie sú reprezentované pointrami na objekty, ale ich identifikátormi. Táto trieda tak poskytuje informácie o najkratšej ceste medzi dvoma uzlami cestnej siete na základe ich identifikátorov. Trieda sa používa nasledujúcim spôsobom

1. Metódou **`Clear`** sa zrušia akékoľvek informácie uložené v triede a môže sa tak začať s výpočtami.
2. Pomocou metódy **`AddCrossroad`** si postupne uloží všetky vrcholy grafu s hranami ktoré z neho vedú.

3. Ďalej sa pomocou metódy **`MakeTable`** postaví tabuľku, nad ktorou bude pracovať algoritmus vyhľadávania minimálnych ciest.
4. Metóda **`ComputePaths`** potom môže spočítať pomocou Floyd-Warshallovho algoritmu najkratšie cesty medzi každými dvoma vrcholmi.

3.2.16. RoutePlan

Trieda **`RoutePlan`** reprezentuje plán cesty automobilu mestom. Jeho prvoradou úlohou je poskytovať automobilu informáciu o tom, ktorým smerom by sa mal vydať. Jeho druhoradou úlohou je zaznamenávanie štatistických dát ako dĺžka trasy a trvanie vykonávania konkrétneho plánu. Plán obsahuje pointre na:

- budovy, kde sa cesta začína a končí
- ich najbližšie cesty
- zoznam uzlov cestnej siete, ktorými prechádzal automobil počas cesty do cieľa

Na zber štatistických dát používa dátové položky **`total_duration_`**, **`mileage_`**, **`times_performed_`**. Metódou **`Begin`** spustí automobil meranie plánu a metódou **`Commit`** ho zastaví a zároveň potvrdí jeho uloženie. Ak náhodou nie je zavolaná metóda **`Commit`** nejakého plánu pri zobrazovaní štatistických dát, údaje z tejto poslednej nevykonanej cesty sa nezobrazujú v celkových štatistikách.

3.2.17. Tester

Obsahuje súbor testovacích scenárov, ktoré sú používané v testovacom móde. Vznikla ďalšia štruktúra **`CarModel`**, ktorá sadou svojich parametrov slúži ako model pri vytváraní objektu Automobil. Táto trieda má len jedinú inštanciu - ako súčasť triedy **`Town`**, ktorá pri načítaní mapy a vytváraní automobilov podľa prítomnosti testovacieho režimu načíta príslušnú mapu a modely áut zo svojho **`testera`**.

3.3. Algoritmy

3.3.1. Algoritmus kroku autíčka:

```
IF auto je zaparkované THEN

        IF auto je doma THEN
                IF zazvonil budík a práca ešte neskončila THEN
                        snaž sa dostať z domu do prace;

        ELSE IF auto je v práci THEN
                IF práca sa skončila THEN
                        IF auto ma vybratý obchod THEN
                                snaž sa dostať z práce do obchodu;
                ELSE snaž sa dostať z práce domov;

        ELSE IF auto je v obchode THEN
                IF nakupovanie skončilo THEN
```

```
                        snaž sa dostať z obchodu domov;

        ELSE chyba: auto nachádza na neznámom mieste;

ELSE //je auto na ceste}
        sleduj cieľ svojej cesty;

aplikuj zmeny polohy na auto;
skontroluj stav a poprípade zahlás chybu;
```

3.3.2. Algoritmus pohybu automobilu v cestnej premávke

```
IF blízko aktuálneho cieľa THEN
     Načítaj údaje o ďalšom uzle cestnej siete;
     IF blízko konca cesty THEN
          IF auto sa pohybuje THEN
               spomaľuj do ciela;
          ELSE zaparkuj;
     ELSE IF blízko zákruty THEN
               prejdi cez zákrutu;
     ELSE IF blízko križovatky THEN
               prejdi cez krizovatku;
     ELSE pokračuj rovno;
```

3.3.3. Algoritmus prechodu automobilu križovatkou

Podmienky na začiatku:

- Najbližší cieľ je križovatka.
- Pri prvom volaní metódy sme v poslednom kroku, kedy automobil dokáže ubrzdiť pred križovatku.

```
IF môžeš vstúpiť do križovatky THEN
        Ignorujem signály križovatky = true;
        oznám ďalšiemu uzlu cesntje siete, že sa onedlho zaradíš do
         fronty smerujúcej k nemu;
         vstúpiš;
        oznám križovatke, že do nej onedlho vstúpiš;
        IF auto bočí na križovatke THEN
                bod := bod v krizovatke, kde sa treba otočiť;
                        cieľový bod := bod;
                        prejdi cez zákrutu; // specifikovanu tymto bodom
                ELSE
                        posuň body trasy;
                        pokračuj rovno; // za novym bodom trasy
ELSE brzdi;
```

3.3.4. Algoritmus prechodu automobilu zákrutou

Podmienky na začiatku:

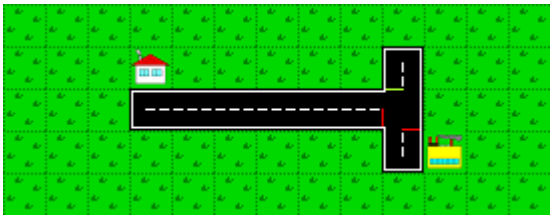
- Pri prvom volaní metódy sme v poslednom kroku, kedy automobil dokáže ubrzdiť pred križovatku.

```
IF aktuálna rýchlosť > bezpečná rýchlosť v zákrute OR za zákrutou je
plno THEN
        brzdi;
ELSE IF aktuálna rýchlosť > vzdialenosť do bodu otočenia THEN
        premiestni auto do bodu otočenia;
        posuň ciele;
        zmeň smer rýchlosti; //auto sa "pootocilo"
        preveď korekciu polohy auta;
ELSE pokračuj rovno;
```

4. Testovanie a testovacie dáta

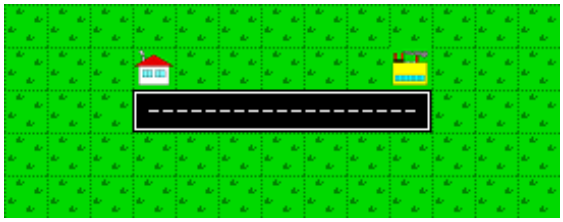
Najdôležitejším predpokladom pre vývoj simulačného programu je kvalitné testovanie. Často sa zlý stav simulácie odhalí iba opticky a dopátrať sa príčiny pri tisícke objektov, ktoré sa v simulácii vyskytujú nie je jednoduché.
Základom pre testovanie je štruktúra s názvom **`tester`** a **`CarModel`**. **`Tester`** obsahuje súbor testovacích scenárov, ktoré sú používané v testovacom móde. Testovací scenár je tvorený mapou a zoznamom automobilov, ktoré ju budú testovať. **`CarModel`** obsahuje všetky parametre automobilu, ako napríklad odkiaľ má začínať svoju cestu, kedy má na ňu vyraziť.
Ako postup na odhaľovanie chýb sa štandardne núka krokovanie a sledovanie premenných vo vývojovom prostredí. Pre účely odhaľovania chýb v simulácii však tieto postupy nevyhovujú v dostatočnej miere. V prvom rade sa chybný stav simulácie dá len ťažko určiť len na základe vedomostí premenných simulovaných objektov. Je nutné celú simuláciu vidieť znázornenú na grafickom výstupe, aby sa jej korektnosť dala posúdiť. Vývojové prostredie počas krokovania ale neumožňuje zobraziť okno simulácie, a teda ani grafiku. Musela tak byť vytvorená alternatíva krokovania a sledovania premenných priamo v programe.
Bola zavedená metóda **`flush`**(), ktorá vypisuje stav objektu na štandardný výstup a metóda **`report`**(), ktorá vypisuje hlásenie od objektu. Do programu bolo pridané tlačidlo Stop, ktorým sa simulácia dá kedykoľvek pozastaviť. Následne sa využije tlačidlo Next, ktorým sa odsimuluje jeden krok. Vhodným umiestnením spomínaných metód do sledovaného kódu sa tak dá v program odkrokovať podobným spôsobom ako sa používa krokovanie vo vývojovom prostredí. Dĺžka jedného simulačného kroku je nastavená na vyššiu hodnotu, aby sa pohyb objektov za spustenej simulácie dal sledovať detailnejšie.
Program umožňuje navyše priamy prístup k objektu zobrazenému na obrazovke jednoduchým kliknutím. Okno získa od triedy **`Town`** pointer na objekt, ktorý sa nachádza na mieste kliknutia Po vybratí objektu sa pomocou tlačidiel dá volať metóda **`flush`**() a nastaviť objektu, či sa má vypisovať na obrazovku metódou **`SetReportMode.`**
V tomto simulačnom programe je z hľadiska testovania najdôležitejšia trieda Automobil, pri ktorej nás zaujíma hlavne jeho pohyb mestom, výber trasy a iné. Ďalej je potrebné otestovať spoluprácu a vzájomnú komunikáciu medzi automobilom a cestami, za pomoci ktorých sa automobil pohybuje po meste. Na otestovanie konkrétnej metódy či situácie sa zvyčajne použil jeden testovací scenár.

4.1. Zoznam testovacích scenárov



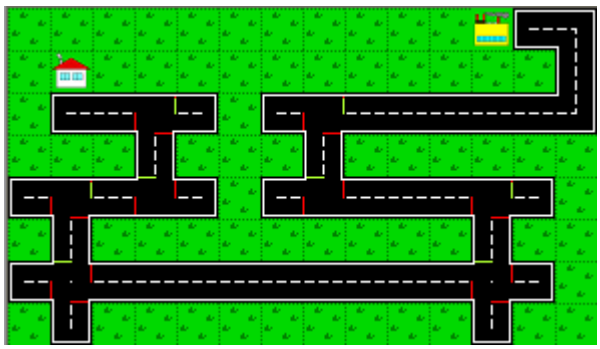
Obrázok č.2 – testovacia mapa č.1

Testovací scenár č.1
Testovaná metóda:
Automobile::GoThroughCrossroad()
Situácia: Testuje sa správanie sa auta, ak musí hneď za križovatkou zísť z cesty.



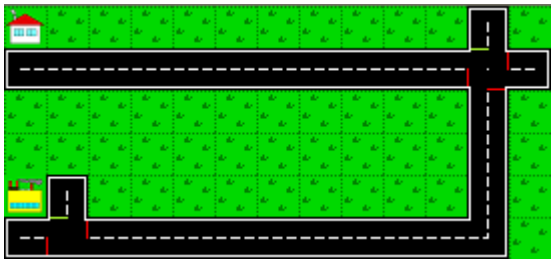
Obrázok č.3 – testovacia mapa č.2

Testovací scenár č.2
Testovaná metóda :
Automobile::GetDirectionsBetween()
Situácia: Špeciálny prípad pre metódu, ktorá nastavovala plán cesty mestom.



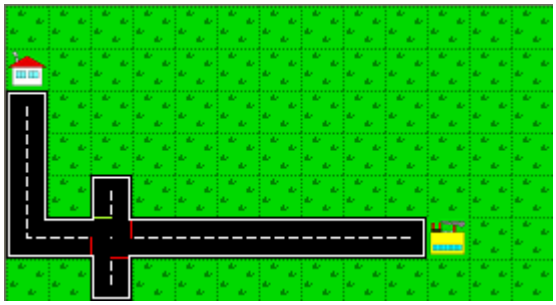
Testovací scenár č.3
Testovaná metóda: Automobile::AdaptSpeed()
Situácia : Prvé auto vyrazí skôr a bude brzdiť druhé.

Obrázok č.4 – testovacia mapa č.3



Testovací scenár č.4
Testované metódy: Automobile::Brake(), Automobile::AcelerateWithLimit(), Automobile::AdaptSpeed(),
Situácia : Či automobil berie dobre do úvahy brzdy auta pred ním. Zaujíma ma najmä situácia, keď idú za sebou dve autá a obe sa blížia ku križovatke. Prvé z nich ma silnejšie brzdy a druhé je hneď za ním.

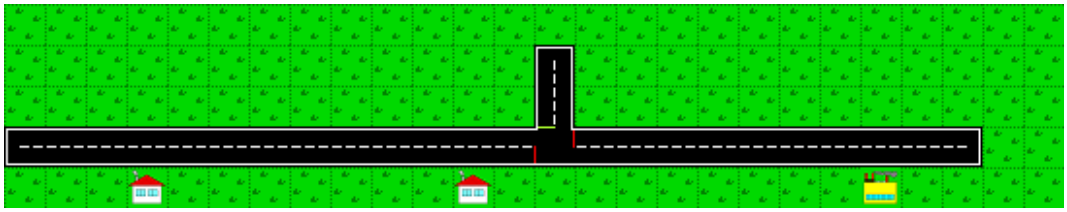
Obrázok č.5 – testovacia mapa č.4



Testovací scenár č.5
Testovaná metóda:
Automobile::GoThroughCrossroad
Situácia : Čo sa stane, ak je za zákrutou plno a do zákruty ide ďalší automobil.

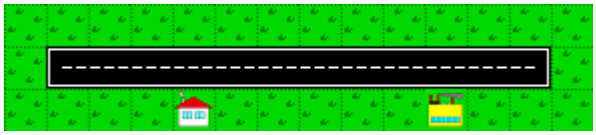
Obrázok č.6– testovacia mapa č.5

Testovací scenár č.6



Obrázok č.7 – testovacia mapa č.6

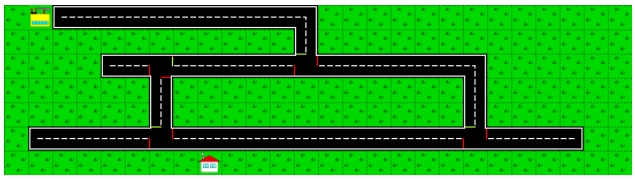
Testovaná metóda: Automobile::EnterRoad()
Situácia: Test na výjazd automobilu na cestu. Pošleme niekoľko automobilov za sebou(4-10) a sledovaný automobil sa bude snažiť bezpečne vyjsť na cestu. Zároveň budeme sledovať, ako vychádzajú automobily za sebou na cestu.



Testovací scenár č.7
Testovaná metóda:
Automobile::GetDirectionsBetween()

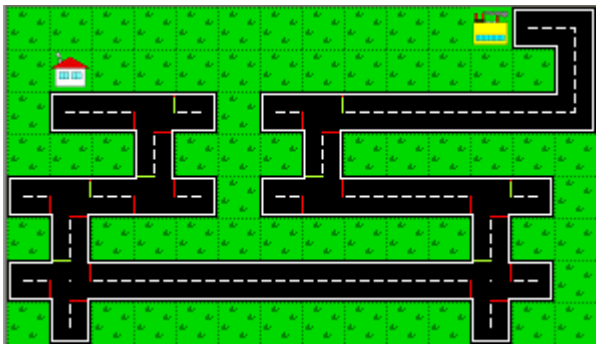
Obrázok č.8 – testovacia mapa č.7

Situácia: Použijeme jeden automobil a jednu jedinú rovnú cestu, kde na náprotivných stranách sa budú nachádzať továreň a dom. Spomínané dva objekty budú mať ako najbližšiu cestu stanovenú buď priamu cestu alebo slepú uličku. Otestujeme všetky možné kombinácie, ktorých je celkovo 8.



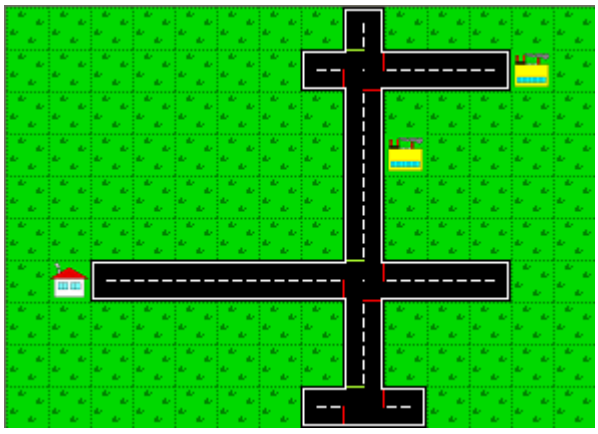
Testovací scenár č.8
Testovaná metóda :
Automobile::GetDirectionsBetween()

Obrázok č.9 – testovacia mapa č.8



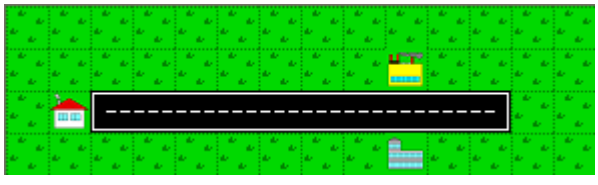
Testovací scenár č.9
Testované metódy :
Automobile::GetDirectionsBetween(),
Crossroad::GetInnerEntryPoint(),
Crossroad::GetInnerExitPoint(),
CrossroadLogic (sada set/get metód) - pridanie programovateľnej logiky.

Obrázok č.10 – testovacia mapa č.9

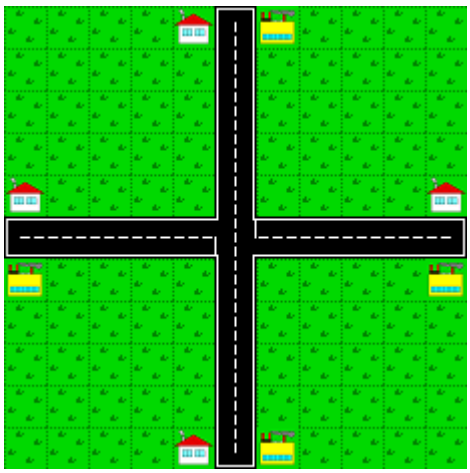


Testovací scenár č.10
Testované metódy:
Automobile::GoThroughCrossroad(),
CarQueue::WillBeFree(),
CarQueue::AddIncommer(),
CarQueue::RemoveIncommer()
Situácia: Testovanie, či autá ostanú stáť pred križovatkou, ak je cesta za križovatkou plná. Test na novo pridané metódy fronty áut.

Obrázok č.11 – testovacia mapa č.10



Testovací scenár č.11
Testovaná metóda: TryStrikeOutOnRoute()
Situácia: Čo sa stane ak sa nachádza práca rovno oproti obchodu? Automobil sa musí presunúť rovno do obchodu



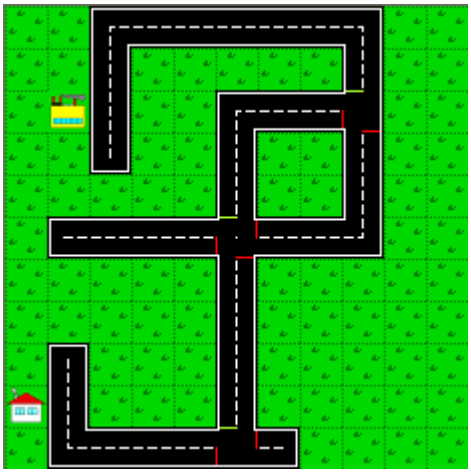
Obrázok č.12 – testovacia mapa č.11

Testovací scenár č.12

Testovaná metóda: Crossroad::GreenForDirectionFrom()

Situácia: Test na načítanie mapy po pridaní možnosti ukladania smeru hlavnej cesty a logiky križovatky Testovanie prechodu automobilu križovatkou s hlavnou ulicou.

Obrázok č.13 – testovacia mapa č.12

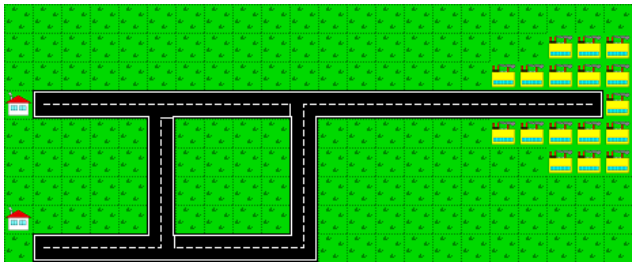


Testovací scenár č.13

Testovaná trieda: RoutePlan()

Situácia: kontrolujeme správne ukladanie sa a tvorbu plánu pri prechode automobilu mestom.

Obrázok č.14 – testovacia mapa č.13

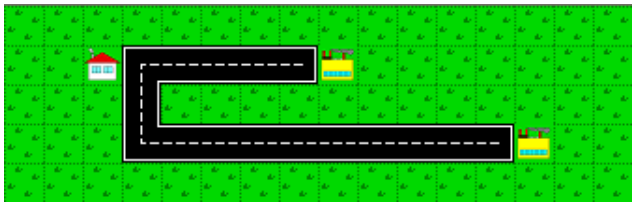


Testovací scenár č.14

Test: pridanie logiky vodiča.

Situácia: Testujeme preplánovanie trasy progresívneho/chaotického vodiča. Po vrchnej ceste je pustené veľké množstvo automobilov, tak aby zablokovali automobily prichádzajúce zdola, ktoré obsadia túto vedľajšiu cestu do požadovaného množstva.

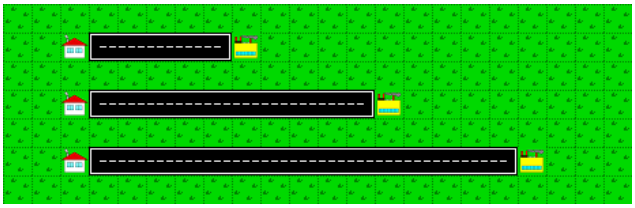
Obrázok č.15 – testovacia mapa č.14



Testovací scenár č.15

Test: Pridávanie novej cesty. Zisťujeme, či ostane zachovaný invariant o križovatkách. Cestu sa snažíme pridať na políčko obkolesené troma políčkami typu cesta.

Obrázok č.16 – testovacia mapa č.15



Obrázok č.17 – testovacia mapa č.16

Testovací scenár č.16
Test: pridanie štatistických údajov

5. Užívateľská dokumentácia

Tento program bol vytvorený s cieľom umožniť užívateľovi simulovať cestnú premávku v meste, upravovať ju požadovaným spôsobom a porovnať efektívnosť jednotlivých správaní sa vodičov v nej. Užívateľ tak môže vyhodnotiť, aký algoritmus voľby trasy mestom (správanie sa vodiča) je z časového hľadiska najefektívnejší a v akej cestnej sieti sa najviac prejavujú jeho pozitívne výsledky. Zároveň je užívateľovi umožnená editácia cestnej siete s úlohou postaviť ju čo najpriechodnejšiu.

5.1. Čo sa v programe simuluje

Tak ako v reálnom živote aj v tejto simulácii každý **človek** niekde býva, pracuje a nakupuje. V prvý deň svojho pobytu v meste sa človek vydáva do práce zavčasu tak, aby do nej stihol dôjsť načas. Po práci sa vydá nakupovať do obchodu. Ak žiaden obchod v meste nie je, putuje človek po práci priamo domov. Z obchodu potom cestuje domov.
Ako **vodič** sa správa vždy podľa pravidiel cestnej premávky. Vždy sa snaží cestovať maximálnou povolenou rýchlosťou, aby sa mestom prepravil do svojho cieľa, čo najrýchlejšie. Všetky automobily v tejto simulácii majú rovnaké jazdné vlastnosti, a tak z predošlého tvrdenia vyplýva, že nikdy nepredbieha automobily idúce pred ním.
Ako **správanie sa** vodiča sa v tomto programe chápe, aký algoritmus výberu cesty mestom volí vodič. Každý vodič môže mať jedno z troch správaní sa

- **Konzervatívne**
  Vodič v prvý deň, keď ešte nemá nastavený plán trasy, ho vytvorí počas prechádzania mestom tak, že na každej križovatke zvolí smer, ktorým vedie do cieľa najkratšia cesta. Po prvom dni už tento plán nikdy nemení, okrem prípadov zmeny cestnej siete užívateľom.

- **Progresívne**
  Toto správanie sa vyznačuje dvoma parametrami:
  - **hladina tolerancie naplnenosti cesty** – udáva maximálny pomer naplnenosti cesty, ktorou sa má vydať, pri ktorom u neho ešte nedochádza k nespokojnosti.
  - **hladina tolerancie vzdialenosti** – predstavuje percentuálne predĺženie alternatívnej cesty oproti minimálnej.

  Vodič sa snaží pred križovatkou preplánovať svoj plán, ak vidí, že cesta ktorou je do cieľa najkratšie má vyššiu naplnenosť ako je jeho hladina tolerancie objemu. V tom prípade volí najkratšiu cestu zo všetkých, ktorou si ale nesmie predĺžiť trasu o viac ako je jeho hladina tolerancie vzdialenosti.

- **Chaotické**
  Vodič na každej križovatke okrem cieľovej volí ako ďalší smer ten, ktorý má najmenší súčin obsadenosti cesty za križovatkou a vzdialenosti cez ňu do cieľa.

V cestnej premávke sa vyskytujú **križovatky**, ktorých hlavnou funkciou je usmerňovanie a regulovanie cestnej premávky. Užívateľ ich v tomto programe môže nastavovať tak, aby dosiahol čo najlepšiu priechodnosť cestnej siete. V tejto simulácii sa vyskytujú križovatky dvoch druhov:

Svetelná križovatka
Logika svetelnej križovatky pozostáva zo 4 krokov, ktoré sa cyklicky menia. Každý krok má

1. dĺžku svojho trvania
2. 4 smery, v ktorých v ktorých sú v tomto kroku prepúšťané automobily križovatkou.

Križovatka s hlavnou cestou
Tento typ cesty je mierne zjednodušene simulovaný oproti realite, ale na účele hlavnej cesty to nič nemení. Použité sú nasledujúce zjednodušené pravidlá:

- Ak ide automobil po hlavnej ceste, je mu vždy umožnený prejazd
- Ak schádza z hlavnej cesty tak, že križuje protiidúci hlavný smer, musí byť tento protismer voľný
- Ak prichádza z vedľajšej cesty s vyššou prioritou, musia byť uvoľnené oba hlavné smery.
- Ak prichádza z vedľajšej cesty s nižšou prioritou, musia byť všetky ostatné smery voľné.

5.2. Popis programu

Pre dosiahnutie cieľov programu, sú užívateľovi poskytnuté nasledujúce funkcie, ktoré sa odporúča vykonávať v tomto poradí:

- výstavba mesta, zónovanie a stavba ciest
- úprava cestnej siete, najmä križovatiek
- nastavovanie parametrov novo generovaných vodičov a ich správania sa
- simulovanie cestnej premávky
- porovnávanie jednotlivých typov správaní sa vodičov
- vyhodnocovanie vlastností postavenej cestnej siete

Hlavné okno programu pozostáva z 3 častí:

- Grafická časť s mapou mesta, v ktorej sa zobrazuje priebeh cestnej premávky počas dňa.
- Ovládací panel, na ktorom sa nachádzajú všetky ovládacie prvky, poskytujúce užívateľovi spomínané funkcie.
- Panel systémových hlásení, kde sa užívateľovi zobrazujú odpovede a hlásenia o priebehu ním vykonanej akcie.
- Pomocné okno čiernej farby, ktoré slúži na zobrazovanie dodatočných informácií.

Obrázok č. 18 - vzhľad hlavného okna programu

5.3. Funkcie ponúkané programom

5.3.1. Výstavba mesta, zónovanie a stavba ciest

Menu výstavby mesta sa zobrazí po kliknutí na záložku **Build-up**. Pred začatím simulácie je potrebné vytvoriť podmienky na príchod obyvateľov do mesta. To znamená, že musí byť vytvorený dostatok nových domov a pracovných miest. Za týmto účelom môžete sledovať informácie o počte obyvateľov a počte voľných domov a pracovných miest môžete sledovať v časti s názvom **Town Characteristics.**

Na výber je v tomto menu z nasledovných akcií:

- **Výstavba budov**
  Kliknite na jedno z tlačidiel **Residental Zone**, **Commercial Zone**, **Industrial Zone** podľa typu budovy ktorú chcete postaviť a kliknite do mapy na políčko trávy, kam chcete túto budovu umiestniť.

- **Výstavba ciest**
  Kliknite na tlačidlo **Road** a kliknite do mapy na políčko trávy kam chcete umiestniť cestu.
  Poznámka : Cesty je možné jedine napájať na už existujúcu cestnú sieť, aby sa zamedzilo vzniku nespojitej cestnej siete. Nesmú nahradiť budovy a môžu ležať na tráve alebo vode.

- **Mazanie políčok**
  Ľubovoľné políčko môžete zmazať tak, že ho nahradíte políčkom trávy alebo vody. Vyberte jedno z tlačidiel **Grass** alebo **Water** a kliknite do mapy na políčko, ktoré chcete nahradiť.
  Poznámka : Cestu je možné mazať/nahradzovať jedine, ak jej vymazaním nevznikne nespojitá cestná sieť.

- **Zmazanie celej mapy**
  Kliknite na tlačidlo **Clear Map** a celá mapa sa pokryje políčkami trávy. Na mape sa objavia aj dve políčka cestnej siete, na ktorú treba napojiť nové cesty.

- **Uloženie mapy**
  Kliknite na tlačidlo **Save Map As** a uložte mapu do Vami zvoleného súboru.

- **Načítanie mapy**
  Pre načítanie mapy musíte najprv vybrať jeden z typov mapy, ktorú chcete načítať. Tento zoznam sa nachádza nad tlačidlom **Load Map**, ktoré slúži na načítanie príslušnej mapy. Po kliknutí na **Load Map** pri vybratej položke „**from external file *.rsh**“ môžete vybrať súbor, z ktorého sa má načítať mapa. Pri vybranom „**test map no. ..**“ a kliknutí na **Load Map** sa načíta testovacia mapa aj s automobilmi, ktoré ju budú testovať.

Poznámka č.1: Výstavba mesta nemôže prebiehať počas simulačného procesu.
Poznámka č.2: Ak je nejaké políčko zóny šedé, znamená to, že v ňom nikto nebýva/nepracuje.

Tip : Dosiahnutie 100% naplnenosti mesta
Pre najväčšiu naplnenosť mesta je potrebné, aby každý terajší alebo budúci obyvateľ mohol mať prácu. Z toho vyplýva, že je potrebné vyrovnať počet rezidentálnych políčok s súčtom industriálnych a komerciálnych.

5.3.2. Nastavovanie križovatiek cestnej siete

Križovatky sa dajú nastavovať po kliknutí na záložku **Crossroads**. Po kliknutí do mapy na križovatku sa zobrazí menu určené pre editáciu križovatky, kde je možné realizovať

- **Zmena režimu križovatky**
  Každá križovatka môže pracovať v dvoch režimoch : svetelná križovatka, križovatka s hlavnou cestou. Medzi týmito režimami sa dá prepínať pomocou tlačidla **Change Mode.** Nezávisle na tom, v akom režime pracuje križovatka sa dajú upravovať nastavenia oboch režimov.

- **Úprava hlavnej cesty**

  - **Upravovanie smeru hlavnej cesty**
    Pre editáciu hlavnej cesty je treba kliknúť na záložku **Main Street**. Pre zadanie smeru hlavnej cesty (odkiaľ kam) je potrebné zaškrtnúť práve dve **zaškrtávacie tlačidlá** na kríži zo zaškrtávacích tlačidiel a smer sa nastaví.

- **Úprava svetelnej križovatky**

  - **Priradenie prednastavenej logiky v svetelnom móde**
    Pri vytváraní cestnej siete sa automaticky vytvárajú svetelné križovatky s prednastavenou logikou. Táto logika sa dá križovatke kedykoľvek znovu nastaviť kliknutím na tlačidlo **Default Logic.** Prednastavená logika svetiel znamená, že sa cyklicky menia kroky, kde v každom kroku sú automobily púšťané do križovatky z jedného smeru a opustiť ju môžu hociktorým smerom. Zaručí sa tak, aby boli všetky smery rovnako často a v rovnakých množstvách vyprázdňované.

  - **Poradie menenia krokov**
    Pre menenie poradia krokov vzostupne zaškrtnite tlačidlo **Clockwise**. Pre menenie poradia krokov zostupne toto tlačidlo odškrtnite.

    Poznámka: Prednastavená logika križovatky sa mení vzostupne a smery sú usporiadané tak, aby sa menili v smere pohybu hodinových ručičiek. T.j. najprv svieti zelená pre automobily idúce zhora, potom sprava a t.ď.

  - **Určenie počiatočného kroku algoritmu križovatky**
    Kliknite na tlačidlo **Step** s poradovým číslom kroku, ktorý chcete aby bol prvý a zaškrtnite tlačidlo **First.**

  - **Určenie presného časového okamihu počiatku algoritmu križovatky**
    Pre nastavenie, v ktorom časovom okamihu algoritmu svetiel križovatky má začať križovatka svoje fungovanie na začiatku dňa použite posuvné tlačidlo **Start Time Offset.**

  - **Dĺžka trvania jedného kroku**
    Kliknite na tlačidlo **Step** s poradovým číslom kroku, ktorému chcete zmeniť dĺžku trvania a pohnite posuvným tlačidlom **Step Duration** na požadovanú hodnotu

  - **Nastavovanie smerov ktorými sú automobily prepúšťané križovatkou vo vybranom kroku**
    - Kliknite na tlačidlo **Step** s poradovým číslom kroku, ktorému chcete meniť smery

- Kliknite na tlačidlo pod nadpisom **Dir**, s číslom smeru, ktorý chcete zmeniť
- Zmeňte smer tak, že odškrtnete všetky zaškrtnuté **zaškrtávacie tlačidlá** kríža smerov
- Následne kliknite na **zaškrtávacie tlačidlo**, ležiace v smere, z ktorého budú automobily púšťané do križovatky a za ním kliknite na **zaškrtávacie tlačidlo**, ktorým majú automobily opustiť križovatku.

Synchronizácia svetelných križovatiek
Na dosiahnutie čo najlepších vlastností cestnej siete sa v reálnom svete používa takzvaná zelená vlna. V tomto programe sa ju môžete pokúsiť vytvoriť za pomoci nastavovania hlavne dĺžky trvania jednotlivého kroku križovatky a počiatočného okamihu križovatky, v ktorom začína svoje fungovanie na začiatku nového dňa. Dĺžky týchto časových okamihov sú rozdelené malými zarážkami, ktorých rozostup je 2 čo značí časový ekvivalent, za ktorý prejde automobil jedno políčko maximálnou rýchlosťou. Pre dosiahnutie zelenej vlny je potrebné, aby automobily smerujúce z najbližšej križovatky dorazili k aktuálnej najskôr vo chvíli, keď sa začne krok, v ktorom majú zelenú. Nastavením dĺžky predošlého kroku tak, aby sa končil v čase, kedy dorazia automobily z vedľajšej križovatky (po násobkoch spomenutého rozostupu, ktoré odpovedajú počtu políčok medzi synchronizovanými križovatkami) sa dosiahne efekt zelenej vlny. Ďalší krok sa však začína až za malý časový okamih (veľkosti 6), potrebný na bezpečné opustenie všetkých automobilov križovatku.

Príklad: Vezmime si prípad kedy sú dve križovatky vzdialené od seba 5 políčok a chceme ich medzi sebou zosynchronizovať. Zistíme si začiatok kroku, v ktorom sa automobily vydajú k druhej zo synchronizovaných križovatiek. K tomuto času prirátame čas, ktorý sa rovná vzdialenosti v políčkach medzi križovatkami vynásobený časom, za ktorý prejde automobil jedno políčko (veľkosť rozostupu = 2). Na tento nový čas (ktorý sa dá odstupňovať zarážkami) nastavíme začiatok kroku druhej križovatky, v ktorom budú automobily z prichádzajúce z prvej križovatky prepustené ďalej.

Poznámka: editácia križovatiek je prístupná len vtedy ak simulačný proces nie je aktívny.

5.3.3. Nastavovanie vlastností novo vygenerovaných automobilov

Po vytvorení podmienok prisťahovania sa ľudí do mesta v menu Biuld-up a nastavení cestnej siete v menu Crossroads je možné nastavovať vlastnosti automobilov, ktoré budú generované do nového simulačného dňa. Toto menu je prístupné po kliknutí na záložku **Generator**. Na výber máte z týchto akcií:

- **Nastavovanie počtu novo vygenerovaných automobilov**
  Počet novo vygenerovaných automobilov sa môže pohybovať v rozmedzí od nuly až do maxima obyvateľov, ktorí sa môžu prisťahovať do mesta. Toto maximum je rovné minimu z voľnej kapacity domov a počtu pracovných miest. Počet novo vygenerovaných automobilov sa zobrazuje v stĺpci **New.** Stĺpec **Total** informuje, koľko automobilov bude v meste po vygenerovaní.

  - **Automatické**
    Po zaškrtnutí tlačidla **use automatic generation** program nastaví počet nových obyvateľov na polovicu z maximálneho počtu ľudí, ktorí môžu prísť do mesta.

  - **Ručné**
    Pri odškrtnutom tlačidle **use automatic generation** sa aktivuje ručné zadávanie novo vygenerovaných automobilov a ich počet je prednastavený na polovicu. Môžete nastaviť

počet automobilov v nasledujúcom dni dvoma spôsobmi:
- zvýšiť – posuňte posuvným tlačidlom **Cars to generate** doprava
- znížiť – posuňte posuvným tlačidlom **Cars to generate** doľava

- **Nastavovanie pomeru novogenerovaných automobilov**
  Pred každým simulačným dňom je možné nastaviť pomer počtov novo vygenerovaných automobilov s konkrétnym správaním. Podľa typu správania sa, ktorému chcete nastaviť pomer posuňte posuvným tlačidlom **Conservative**, **Progressive** alebo **Chaos** doprava pre zvýšenie pomeru alebo doľava pre zníženie pomeru.

- **Príklad:** pre generovanie automobilov so správaním sa conservative : progresive v pomere 50:50 nastavte jednotlivé posuvné tlačidlá takto : Conservative = 100%, Progressive = 100%

- **Zmazanie všetkých automobilov**
  Kliknite na tlačidlo **Delete All Cars** a vymažú sa všetky automobily v simulácii. Zároveň sa nastaví počet dní simulácie na 0.

- **Nastavenia progresívneho správania sa vodičov**
  Progresívne správanie je charakterizované dvoma parametrami
- **hladina tolerancie obsadenosti cesty** – maximálna naplnenosť cesty, pri ktorej sa vodič ešte drží starej trasy.
- **hladina tolerancie vzdialenosti** – maximálne predĺženie, ktorým je ochotný sa vydať pri zavrhovaní starej cesty

- **Nastavenie hladiny tolerancie obsadenosti cesty a hladiny tolerancie vzdialenosti**
  - Podľa typu parametra, ktorý chcete nastaviť zvoľte posuvné tlačidlá buď pod označením **Volume tolerance,** ktoré označuje hladinu tolerancie obsadenosti alebo pod **Distance Tolerance,** ktoré označuje hladinu tolerancie vzdialenosti.
  - Novo vygenerované automobily s progresívnym správaním budú mať nastavenú požadovanú hladinu náhodne v rozsahu od hodnoty **Lowest** po hodnotu **Highest.** Pre nastavenie spodnej hranice hladiny použite posuvné tlačidlo **Lowest** a pre nastavenie hornej hranice tlačidlo **Highest**.

  Poznámka : bližší význam oboch parametrov nájdete v časti subkapitoly **Čo sa v programe simuluje.**

5.3.4. Simulácia cestnej premávky

Premávku je možné simulovať po kliknutí na záložku **Simulation**. V tomto menu program ponúka na nasledujúce funkcie:

- **Začatie simulácie**
  Kliknite na tlačidlo **Start** a v grafickej časti môžete sledovať priebeh simulácie.

- **Pozastavenie simulácie**
  Kliknite na tlačidlo **Stop** a simulačný proces sa zastaví.

- **Pokračovanie pozastavenej simulácie**
  Kliknite na tlačidlo **Start** a simulačný proces bude pokračovať

- **Ukončenie simulácie**
  Kliknite na tlačidlo **Sleep** a celá simulácia sa ukončí, všetci obyvatelia, ktorí sa

nenachádzajú vo svojom dome sa do neho premiestnia, tak aby boli pripravení na ďalší simulačný deň.

- **Odsimulovanie jedného simulačného kroku**
  Pri pozastavenej simulácii kliknite na tlačidlo **One Step**

- **Voľba zobrazovania simulácie**
  - **zobrazenie pohybujúcich sa automobilov** - kliknite na tlačidlo **Cars.**
  - **zobrazenie objemu dopravy podľa jednotlivých správaní sa** - kliknite na jedno z tlačidiel **All types, Conservative, Progressive** alebo **Chaos** odpovedajúce typu správania sa, ktorého štatistické údaje chcete zobraziť. Po zakliknutí jedného z nich sa každý úsek cesty zobrazí podľa počtu automobilov, ktoré ním prešli vyplnený jednou z farieb
    - čierna – neprešli ním žiadne automobily zadaného typu
    - žltá – prešlo ním malé množstvo automobilov
    - oranžová – prešlo ním stredné množstvo automobilov
    - červená – prešlo ním veľké množstvo automobilov

- **Zobrazenie detailu objektu v simulácii**
  Po kliknutí myšou na zvolený objekt sa v oblasti **Object Info** zobrazia jeho detaily.

- **Výpis stavu objektu do pomocného okna**
  Po vybratí objektu kliknite na tlačidlo **Flush.** Objekty ktoré vedia vypisovať svoj stav do pomocného okna sú automobil, križovatka, zákruta a slepá ulička.
- **Nastavenie hlásenia akcií vykonávaných objektom do pomocného okna**
  Po vybratí objektu kliknite na tlačidlo **Report** a objekt začne hlásiť vykonávané akcie. Ak máte ešte vybraný stále ten istý objekt jeho hlásenia sa vypnú opätovným kliknutím na tlačidlo **Report**

5.3.5. Vyhodnotenie správaní sa vodičov a vlastností cestnej siete

Po kliknutí v záložke **Simulation** na tlačidlo **Compare Behavoiurs** a v novom okne sa zobrazí tabuľka so štatistickými údajmi, pomocou ktorých sa dajú porovnanie jednotlivé typy správaní sa ako aj celkové vlastnosti postavenej cestnej siete. Toto okno umožňuje uloženie dát, ktoré sa v ňom zobrazujú pomocou tlačidla **Save.** Tieto dáta sa uložia do súboru s názvom „**comparison.txt**“, ktorý nájdete v adresári s programom.

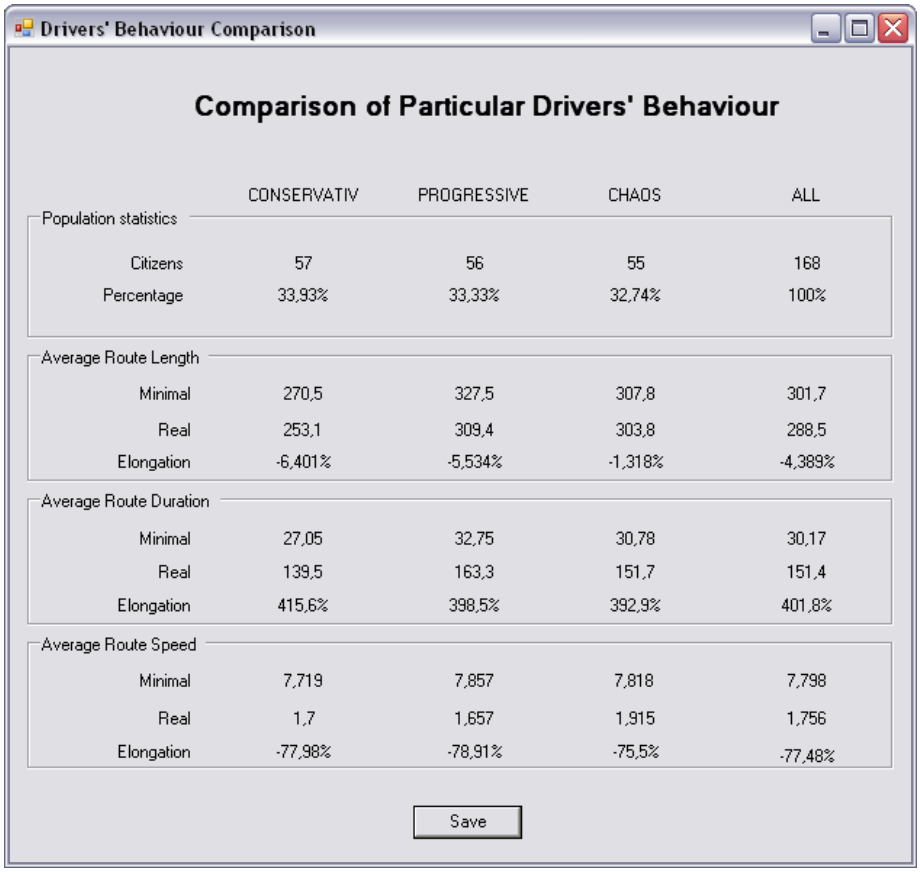
Význam jednotlivých riadkov tabuľky:

1. **Citizens** – počet obyvateľov žijúcich v meste
2. **Percentage** – percento obyvateľov
3. **Average Route Length** – priemerná dĺžka cesty
4. **Average Route Duration** – priemerný čas vykonávania jednej cesty
5. **Average Route Speed** – priemerná rýchlosť cesty automobilu mestom
6. **Minimal** – minimálna hodnota parametru trasy, odvíjajúca sa od cesty v cestnej sieti
7. **Real** – reálna hodnota parametru
8. **Elongation** – percentuálne vyjadrenie rozdielu reálnej hodnoty a minimálnej resp. maximálnej

Najdôležitejším parametrom, ktorý udáva efektívnosť, či už správania sa konkrétneho typu vodiča alebo cestnej siete je parameter **Average Route Speed Real,** udávajúci reálnu priemernú rýchlosť

cestovania mestom.

Tip: Mazanie štatistických údajov
Štatistiky za každý jednotlivý objekt udávajú hodnotu parametra za celé obdobie jeho života v meste. Preto po pri vysokom počte automobilov, ktoré sú dlhší čas v meste, sa zmeny cestnej siete neprejavia výrazne. Na výraznejšie sledovanie vplyvu zmien cestnej siete na jej vlastnosti treba zmazať všetkých obyvateľov mesta tlačidlom **Delete All Cars** v hlavnom okne programu a nagenerovať nových s rovnakými parametrami. Týmto spôsobom sa vynulujú štatistické dáta.



Obrázok č.19 - Okno s porovnaním jednotlivých druhov správaní sa

6. Analýza cieľov projektu

6.1. Porovnanie jednotlivých správaní sa vodičov

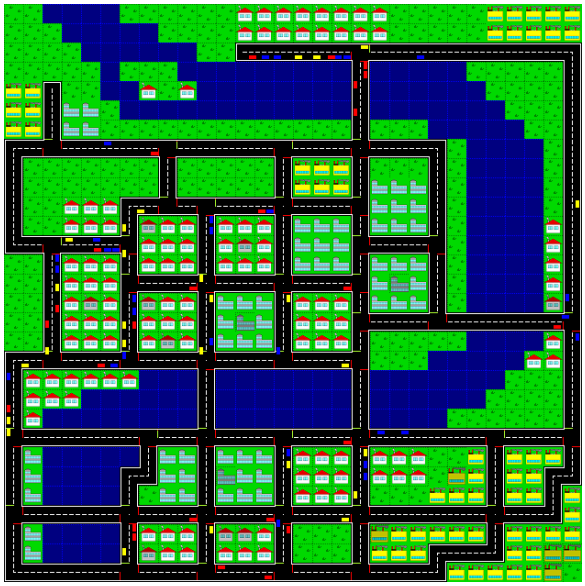
Ako už v úvode bolo spomenuté jedným z hlavných cieľov programu je skúmať, ktoré správanie sa vodiča je počas špičky najúspešnejšie. Jednotlivé typy správania sa sú v grafickom znázornení simulácie rozlíšiteľné podľa farby:

- Konzervatívne – žltá farba
- Progresívne – modrá farba
- Chaotické – červená farba

6.1.1. Vyhodnotenie najefektívnejšieho správania sa

Základnou otázkou pri vyhodnocovaní najefektívnejšieho správania sa je ako objektívne vybrať to najlepšie. Pretože za určitých podmienok môže byť niektoré správanie sa oproti ostatným zvýhodnené. Za týmto účelom boli vyvinuté rôzne konfigurácie simulácie, ktoré by mali zaručiť čo najväčšiu objektívnosť.
Ako základ pre vyhodnocovanie budú použité nasledujúce tri mapy s pevne danou cestnou sieťou a zástavbou.

1. Mapa s nepravidelnou cestnou sieťou.
2. Mapa so štvorcovou cestnou sieťou (rozostupy tak, aby bola dosiahnutá aspoň minimálna zelená vlna)



Obrázok č.20 – Mapa č.1

Obrázok č.21 – Mapa č.2

Zástavba v meste bude v pomere rezidentálna : industriálna : komerciálna 2:1:1. Rozmiestnená bude tak, aby sa čo najviac priblížila realite. Komerciálna zóna bude viac v centre, industriálna viac na okraji mesta. Rezidentálna bude rozložená po celom meste, avšak viac na okrajoch.
Na každej mape je určitý (pevne daný) počet križovatiek. Nastavenie týchto križovatiek taktiež môže ovplyvňovať stav na cestách a poskytnúť tak výhodu niektorému správaniu sa. Preto sú z každej mapy vytvorené 3 varianty, ktorých sa budú nachádzať:

A. Len svetelné križovatky
B. Len hlavné cesty
C. Svetelné križovatky a hlavné cesty v rovnakom pomere

Hlavné cesty aj križovatky budú podľa možností nastavené tak, aby umožnili čo najplynulejšiu premávku mestom.
Ďalším faktorom ovplyvňujúcim úspešnosť niektorých správaní sa môže byť počet automobilov v simulácii. Simulovať sa preto bude niekoľko simulačných dní a každý deň pribudne do simulácie polovica Automobilov. Na konci sledovania bude mesto plne obsadené obyvateľmi.
A nakoniec každú mapu budú simulovať 4 sady automobilov s nasledujúcimi kombináciami správaní sa:

- konzervatívne a progresívne
- progresívne a chaotické
- chaotické a konzervatívne
- konzervatívne a progresívne a chaotické

Kde v každej sade budú typy správaní sa zastúpené v rovnakom pomere.

6.1.2. Výsledky porovnávania

Správanie sa s najväčšou priemernou rýchlosťou v sledovanej konfigurácii je označené ako jej víťaz. Typ správania sa s najväčším počtom víťazstiev je považované za najefektívnejšie spomedzi sledovaných.

Mapa	Typy sledovaných správaní sa							
	Konzervatívne Progresívne		**Progresívne Chaotické**		**Chaotické Konzervatívne**		**Konzervatívne Progresívne Chaotické**	
	víťaz	rýchlosť	víťaz	rýchlosť	víťaz	rýchlosť	víťaz	rýchlosť
Mapa 1-A	P	1,73	C	1,42	K	1,47	P	1,61
Mapa 1-B	P	2,55	C	2,65	C	2,66	C	2,62
Mapa 1-C	P	3,04	C	2,99	C	3,59	K	3,65
Mapa 2-A	P	1,59	C	1,65	C	1,63	P	1,67
Mapa 2-B	K	2,18	C	2,24	C	2,22	C	2,34
Mapa 2-C	K	3,46	P	3,50	K	3,52	K	3,60

Tabuľka č.1 – Výsledky porovnávania správaní sa

Vysvetlivky:

- **K** = konzervatívne správanie sa
- **P** = progresívne správanie sa
- **C** = chaotické správanie sa

Konečné poradie podľa počtu jednotlivých víťazstiev:

1. chaotické (11 víťazstiev)
2. progresívne (7 víťazstiev)
3. konzervatívne (6 víťazstiev)

6.1.3. Rozbor výsledkov

Konzervatívne správanie sa
Voľba minimálnej cesty týchto vodičov sa odvíja od vlastností Bellman-Fordovho algoritmu použitého na jej výpočet. Tento algoritmus v nemodifikovanej podobe poskytuje informácie o jednej minimálnej ceste (nie o všetkých) medzi dvoma vybranými vrcholmi. Z konkrétnej implementácie v tomto programe vyplýva, že ak z križovatky vedú dve rovnako dlhé cesty ku ďalšej križovatke, konzervatívny vodič použije (na základe výsledkov algoritmu) najprv cestu vedúcu hore, vpravo, dole a nakoniec naľavo (z pohľadu grafického zobrazenia v okne programu).
Teda vodiči s týmto správaním sa majú tú vlastnosť, že zahlcujú určité cesty viac ako iné. Vo veľkej miere prispievajú k tvorbe dopravnej zápchy a vytvárajú tak podmienky pre vyniknutie ostatných správaní sa.
Táto vlastnosť sa prejavuje viac na mapách s vyšším počtom križovatiek a niekedy vyústi do nechcenej situácie uviaznutia. Jej príklad je v mape č.2 variant A. Nastane situácia kedy sa v strednej štvorici križovatiek vytvorí zápcha, ktorá sa nemôže skončiť. Na začiatku fronty pred križovatkou je konzervatívny vodič, ktorý nemôže vstúpiť do cesty za križovatkou, ktorá je plná.

Progresívne správanie sa
Snaží sa vyhýbať cestám, ktoré sú už preplnené automobilmi. Ak sú alternatívy príliš neprijateľné vyberie si podľa neho najlepšiu možnosť. Má „pamäť“ v podobe plánu, ktorý vykonal. Teda sa zo skúsenosti by sa mal vyhýbať zlým cestám. Táto pamäť mu však niekedy bráni v skúšaní nových ciest, ktoré by mohli byť nakoniec z časového hľadiska kratšie. Vodiči s vyššou hladinou tolerancie vzdialenosti a nižšou hladinou tolerancie objemu sa viac približujú chaotickému správaniu sa a „experimentujú“ na ceste viac. Progresívny vodič, ktorý má tieto vlastnosti opačné sa zase približuje konzervatívnemu. Vhodnou voľbou týchto hladín sa dosiahne vytvorenie najefektívnejšieho vodiča.

Chaotické správanie sa
Chaotické správanie sa je charakterizované slabou predvídateľnosťou. Riadi sa jedine aktuálnou situáciou na ceste. Niekedy sa tomuto vodičovi oplatí riskovanie. Stane sa, že aj keď si vzdialenosť do cieľa predĺži, čas strávený na tejto alternatívnej ceste sa skráti, pretože je tam lepšia cestná sieť. Chaotik sa tak vyhne zápche a ak sú cesty okolo miest s vysokou hustotou dopravy dobre napojené a usmernené ponúkajú možnosť vyhnúť sa zápche. Chaotik má tendenciu k týmto cestám smerovať, ale nemá pamäť, ktorá by mu jeho konanie uľahčila.
Chaotický človek prichádza do oblasti s továrňami ako prvý. Značí to, že na začiatku keď sú ešte cesty relatívne prázdne, volí najkratšiu alternatívnu cestu. Ako sa cesty postupne zapĺňajú stáva sa, že blúdi po meste a niekedy zájde aj do slepej uličky.

6.2. Vyhodnotenie efektivity cestnej siete

Na rovnakých konfiguráciách ako v predošlej podkapitole môžeme odsledovať, v ktorej cestnej sieti sa automobily dokážu pohybovať, čo najrýchlejšie.

Mapa	Typy sledovaných správaní sa				Priemer
	Konzervatívne Progresívne	Progresívne Chaotické	Chaotické Konzervatívne	Konzervatívne Progresívne Chaotické	
Mapa 1-A	1,73	1,31	1,41	1,59	1,55
Mapa 2-A	1,57	1,60	1,62	1,58	
Mapa 1-B	2,50	2,62	2,66	2,54	2,38
Mapa 2-B	2,14	2,22	2,18	2,17	
Mapa 1-C	3,03	2,88	3,50	3,65	3,37
Mapa 2-C	3,40	3,44	3,51	3,53	

Tabuľka č.1 – Priemerná rýchlosť automobilov na sledovaných konfiguráciách

6.2.1. Rozbor výsledkov

Podľa uvedenej tabuľky sa dá odvodiť pravidlo: čím má mapa vyššie percento svetelných križovatiek, tým má nižšiu premernú rýchlosť automobilov. Pramení to čiastočne aj z obtiažneho synchronizovania svetelných križovatiek, ktoré miesto toho, aby usmerňovali premávku, ju v niektorých prípadoch zbytočne brzdia.
Hlavné cesty majú oproti križovatkám tú výhodu, že doprava cez ne sa riadi podľa aktuálnej situácie na ceste. Automobily nemusia čakať, kým dostanú zelenú, ale akonáhle je možný prejazd križovatkou, využijú to. Vyprázdňujú sa tak rýchlejšie a ako svetelné križovatky. Hlavné cesty slúžia viac na priepustnosť cestnej siete.
Svetelné križovatky na druhej strane slúžia viac reguláciu dopravy. Mali by sa teda mali používať uvážene a hlavne tam, kde by hlavná cesta ignorovala automobily prichádzajúce z vedľajších ciest, na ktoré by sa nedostal rad.

7. Záver

Autor si je vedomý, že téma sa dala spracovať komplexnejšie. Program by sa dal rozšíriť napríklad tak, že chaotické správanie by pripisovalo význam jednotlivým zložkám. Možné by bolo aj doplnenie inteligentného správania sa, ktoré by na výpočet minimálnej cesty nepoužívalo vzdialenosti, ale vyťaženosť ciest za posledné obdobie. Konzervatívny vodič by sa dal napríklad doplniť o výber náhodnej minimálnej cesty. Program sa dá rozšíriť o viac preddefinovaných typov logík semaforu.
Z programátorského hľadiska by som prácu na tomto projekte zhodnotil ako náročnú. Postupne sa projekt rozrástol na veľký, bolo vytvorené množstvo kódu, ktorého napísanie a hlavne otestovanie zabralo veľa času. Počas tohto projektu som sa však naučil osvojiť si techniky z Code Complete [4] a 101 programovacích technik [5], ktoré výrazne prispeli k sprehľadneniu a lepšej orientácii v kóde.
Ako vývojový nástroj bol použitý program Visual Studio 2005 s akademickou licenciou. Na spravovanie dokumentácie v kóde a jej extrakciu z neho som použil voľne stiahnuteľný nástroj Doxygen.
Možné využitie poznatkov získaných z tohto programu je v GPS navigácii automobilov pri plánovaní trasy automobilu mestom. Táto simulácia však nezodpovedá presne realite, ale ako námet na ďalšie rozvinutie stojí za povšimnutie.

8. Zoznam použitej literatúry

[1] Otvorená internetová encyklopédia wikipédia (2008):
http://en.wikipedia.org/wiki/Traffic#Rush_hour

[2] Töpfer P.: Algoritmy a programovací techniky, Prometheus

[3] Otvorená internetová encyklopédia wikipédia (2008):
http://en.wikipedia.org/wiki/Managed_Extensions_for_C%2B%2B

[4] McConnell S.: Code Complete: A Practical Handbook of Software Construction, Microsoft Press

[5] Alexandrescu A., Sutter H.: C++ 101 programovacích technik, Olympia

9. Zoznam príloh na CD nosiči

- Text bakalárskej práce v elektronickej podobe **Bakalárska práca.pdf**
- Inštalačný súbor **RushHour.zip**
- Inštalačná príručka **Instalation Manual.pdf** k inštalačnému súboru
- Užívateľská príručka k aplikácii **User manual.pdf**
- Zdrojové súbory ako súčasť projektu vývojového nástroja MS Visual Studio 2005 v adresári **Project**
- Programová dokumentácia v html formáte v adresári **Documentation**